Main_code

October 20, 2024

1 Structure of the Data

2 Libraries

[ ]:
```
%pip install shap
```

```
Collecting shap
  Downloading shap-0.44.0-cp310-cp310-
manylinux_2_12_x86_64.manylinux2010_x86_64.manylinux_2_17_x86_64.manylinux2014_x
86_64.whl (533 kB)
                           533.5/533.5
kB 3.4 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-
packages (from shap) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from shap) (1.11.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (from shap) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(from shap) (1.5.3)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.10/dist-
packages (from shap) (4.66.1)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/dist-
packages (from shap) (23.2)
Collecting slicer==0.0.7 (from shap)
  Downloading slicer-0.0.7-py3-none-any.whl (14 kB)
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages
(from shap) (0.58.1)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-
packages (from shap) (2.2.1)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in
/usr/local/lib/python3.10/dist-packages (from numba->shap) (0.41.1)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->shap) (2023.3.post1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
```

```
packages (from scikit-learn->shap) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn->shap) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.1->pandas->shap) (1.16.0)
Installing collected packages: slicer, shap
Successfully installed shap-0.44.0 slicer-0.0.7
```

[ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
from tqdm.notebook import tqdm, trange
import time
import seaborn as sns
from scipy.stats import skew, zscore


# SCIKIT - LEARN
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsRegressor
from sklearn.decomposition import PCA


# PYTORCH (for neural networks)
import torch
from torch.utils.data import TensorDataset, DataLoader, random_split, Subset
import torch.optim as optim
from torchvision import transforms
import torchvision.models as models
import torch.nn as nn

# SHAP
import shap



# XGBoost
import xgboost as xgb
from sklearn.linear_model import ElasticNet
```

2.1 Connexion to google drive

```
[ ]: from google.colab import drive
     drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

2.2 Import home-made classes

```
[ ]: !cp '/content/drive/MyDrive/ML BigProject/library.py' /content/
     from library import preprocess_Ad_Table_Trim, DVMdataset
```

Both processing classes will be in the PY file

3 Data importation

```
[ ]: # For classical data
     Ad_table = pd.read_csv('/content/drive/MyDrive/ML BigProject/data/Ad_table␣
      ↪(extra).csv')
     Trim = pd.read_csv('/content/drive/MyDrive/ML BigProject/data/Trim_table.csv')

     # For images
     Price_table = pd.read_csv("/content/drive/MyDrive/ML BigProject/data/
      ↪Price_table.csv")
     Images_table = pd.read_csv("/content/drive/MyDrive/ML BigProject/data/
      ↪Image_table.csv", sep = ',')
     autosWithNames = pd.read_csv("/content/drive/MyDrive/ML BigProject/data/
      ↪autosWithNames.csv", sep=",")


     # Run the hand-made preprocessing class
     preprocess_classical_data = preprocess_Ad_Table_Trim(Ad_table, Trim)
```

4 Constant parameters

```
[ ]: RANDOM_STATE = 999

     # Columns we decided to remove, after the Descriptive statistics part
     columns_to_drop = ['Maker',
                        'Genmodel',
                        'Genmodel_ID',
                        'Adv_ID',
                        'Adv_year',
                        'Adv_month',
                        'Reg_year',
                        'Annual_Tax',
```

```
                    'Color',

                    "Body_type_0",
                   "Body_type_2",
                   "Body_type_3",
                   "Body_type_10",
                   "Body_type_11",
                   "Body_type_12",
                   "Body_type_13",
                   "Body_type_16",
                   "Body_type_17",
                   "Gear_box_2",
                   "Gear_box_3",
                   "Fuel_type_2",
                   "Fuel_type_3",
                   "Fuel_type_11",
                   "Fuel_type_12",
                   "Fuel_type_13"]
```

5 Descriptive statistics

[ ]:
```
y, X = preprocess_Ad_Table_Trim(Ad_table, Trim).final_set(
                one_hot_encoder = False,
                standardization = False,
                remove_outliers = False)
```

[ ]:
```
df = pd.concat([y, X], axis=1)
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['Price', 'Body_type', 'Runned_Miles', 'Engin_size', 'Gear_box',
       'Fuel_type', 'Engine_power', 'Wheelbase', 'Height', 'Width', 'Length',
       'Average_mpg', 'Top_speed', 'Seat_num', 'Door_num', 'Gas_emission'],
      dtype='object')
```

[ ]:
```
df.shape
```

[ ]:
```
(213272, 16)
```

[ ]:
```
df.describe()
```

[ ]:
```
                Price      Body_type  Runned_Miles     Engin_size  \
count  213272.000000  213272.000000  2.132720e+05  213272.000000
mean    12729.713507       9.354266  5.143895e+04       1.879585
std     18565.901874       3.655172  4.252963e+04       0.756648
min       100.000000       1.000000  2.000000e+00       0.600000
25%      4275.000000       7.000000  1.700000e+04       1.400000
```

```
50%      8000.000000       7.000000  4.460750e+04       1.600000
75%     14590.000000      14.000000  7.900000e+04       2.000000
max    999999.000000      18.000000  6.363342e+06       6.800000

            Gear_box      Fuel_type   Engine_power      Wheelbase  \
count  213272.000000  213272.000000  213272.000000  213272.000000
mean        0.669206       5.008079     147.176866    2591.715143
std         0.470500       3.963576      81.839889     393.678023
min         0.000000       0.000000      38.000000       0.000000
25%         0.000000       1.000000      99.000000    2511.000000
50%         1.000000       7.000000     123.000000    2630.000000
75%         1.000000       9.000000     173.000000    2746.000000
max         1.000000      10.000000     740.000000    5246.000000

              Height          Width         Length    Average_mpg  \
count  213272.000000  213272.000000  213272.000000  213272.000000
mean     1533.380500    1890.472148    4346.581164      52.016842
std       129.534593     151.379024     411.927447      20.709752
min         0.000000    1475.000000       0.000000       9.400000
25%      1458.000000    1775.000000    4062.000000      42.800000
50%      1495.000000    1877.000000    4362.000000      51.400000
75%      1620.000000    2019.000000    4663.000000      60.100000
max      2660.000000    2690.000000    6165.000000     471.000000

           Top_speed       Seat_num       Door_num   Gas_emission
count  213272.000000  213272.000000  213272.000000  213272.000000
mean      122.045661       3.934647       3.404530     155.494818
std        18.337492       0.959234       0.982735      43.320758
min        11.000000       0.000000       1.000000       0.000000
25%       109.000000       4.000000       3.000000     129.096431
50%       118.000000       4.000000       4.000000     142.786356
75%       131.000000       4.000000       4.000000     170.643238
max       226.000000      10.000000       7.000000     496.842105
```

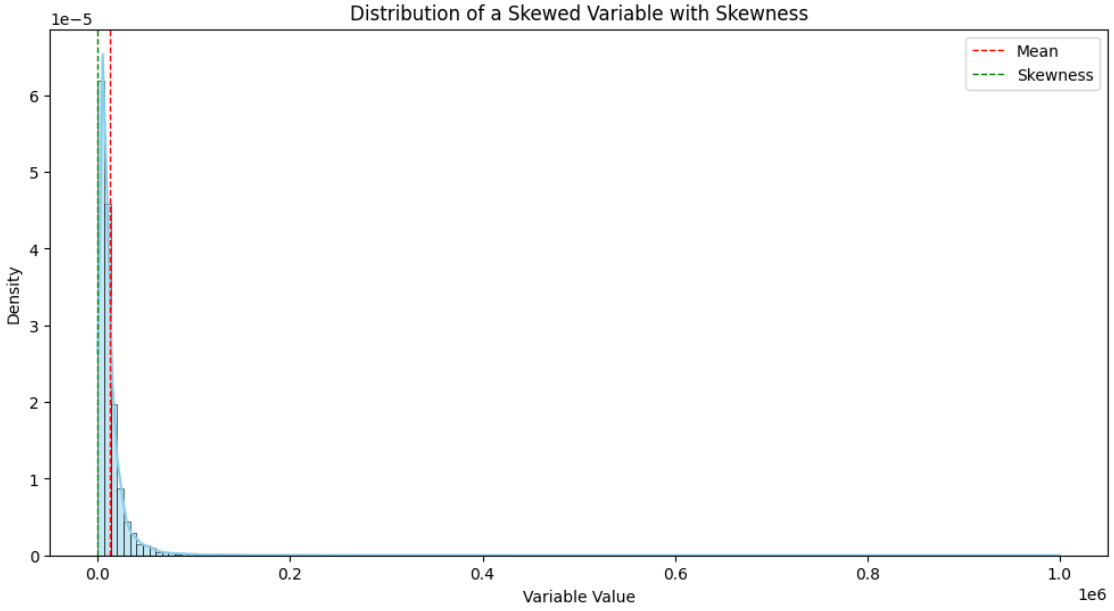
5.1 Analysis of the outliers

5.1.1 Outliers on the target variable, the price

[ ]:
```
y, X = preprocess_Ad_Table_Trim(Ad_table, Trim).final_set(
                one_hot_encoder = False,
                standardization = False,
                remove_outliers = False)
```

[ ]:
```
skewness = skew(df['Price'])

plt.figure(figsize=(12, 6))
sns.histplot(df.Price, bins=150, kde=True, color='skyblue', stat='density')
```

```
# We add vertical lines for mean and the skewness
plt.axvline(np.mean(df.Price), color='red', linestyle='dashed', linewidth=1,␣
 ↪label='Mean')
plt.axvline(skewness, color='green', linestyle='dashed', linewidth=1,␣
 ↪label='Skewness')

plt.title('Distribution of a Skewed Variable with Skewness')
plt.xlabel('Variable Value')
plt.ylabel('Density')
plt.legend()
plt.show()

print(f"Skewness: {skewness}")

"""sns.displot(df, x="Price", color='#e3543b')
plt.title('Distribution of Prices')
plt.xlabel('Price')
plt.show()"""
```




```
Skewness: 8.191398017328405
```

[ ]:
```
'sns.displot(df, x="Price", color=\'#e3543b\')\nplt.title(\'Distribution of
Prices\')\nplt.xlabel(\'Price\')\nplt.show()'
```

[ ]:
```
df.sort_values(by='Price', ascending=False).head(10)
```

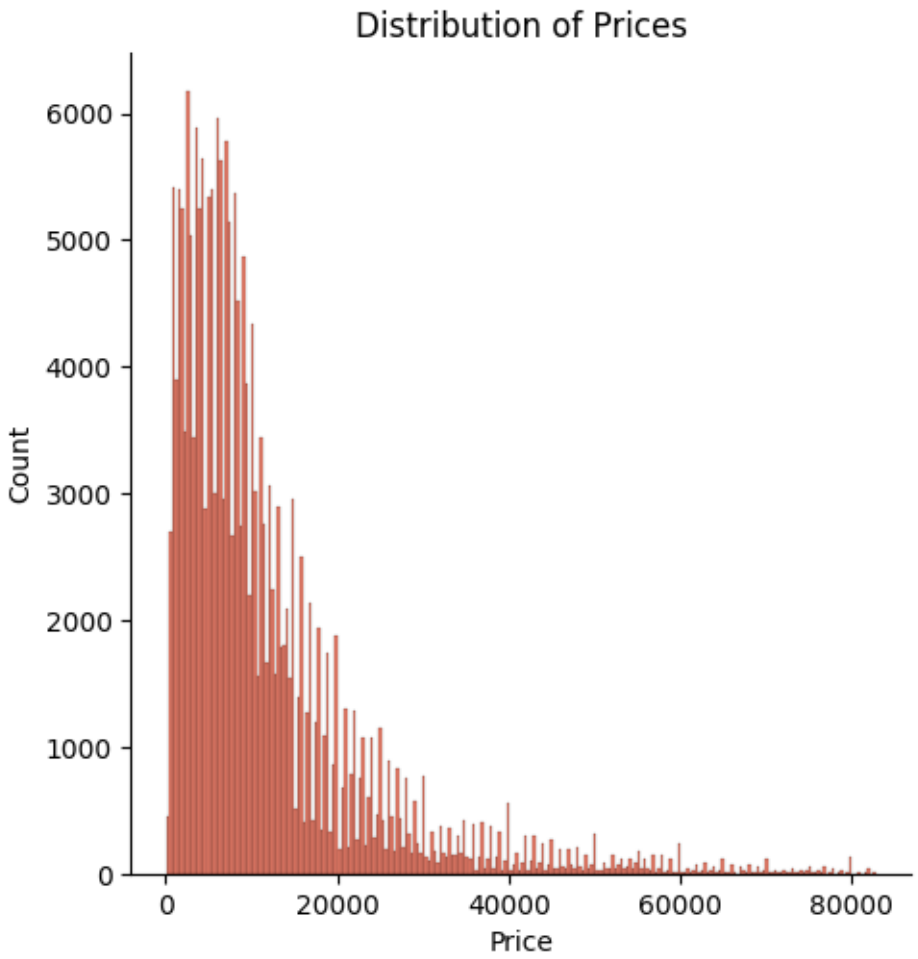
[ ]:

	Price	Body_type	Runned_Miles	Engin_size	Gear_box	Fuel_type	\
170059	999999.0	5	5000.0	5.2	0	9	
14512	625000.0	4	85.0	4.5	0	9	
143496	549995.0	5	50.0	3.8	0	9	
143628	539995.0	5	142.0	3.8	0	9	
144737	535000.0	5	37.0	3.8	0	9	
143318	530000.0	5	100.0	3.8	0	9	
143813	499950.0	5	150.0	3.8	0	9	
154761	495000.0	8	14.0	6.7	0	9	
78997	495000.0	4	2000.0	6.5	0	9	
144304	459900.0	5	25.0	3.8	0	9	

	Engine_power	Wheelbase	Height	Width	Length	Average_mpg	\
170059	602.00	2650.0	1240.0	2037.0	4426.0	23.0	
14512	562.00	2650.0	1213.0	1937.0	4527.0	23.9	
143496	690.62	2453.0	1297.0	1978.0	4549.0	23.9	
143628	690.62	2453.0	1297.0	1978.0	4549.0	23.9	
144737	690.62	2453.0	1297.0	1978.0	4549.0	23.9	
143318	690.62	2453.0	1297.0	1978.0	4549.0	23.9	
143813	690.62	2453.0	1297.0	1978.0	4549.0	23.9	
154761	563.00	3552.0	1646.0	2018.0	5762.0	20.3	
78997	740.00	2700.0	1136.0	2030.0	4835.0	17.6	
144304	690.62	2453.0	1297.0	1978.0	4549.0	23.9	

	Top_speed	Seat_num	Door_num	Gas_emission
170059	205.0	1	1	322.861446
14512	199.0	1	1	289.222222
143496	211.0	1	1	247.303931
143628	211.0	1	1	247.303931
144737	211.0	1	1	247.303931
143318	211.0	1	1	247.303931
143813	211.0	1	1	247.303931
154761	155.0	4	3	363.077922
78997	217.0	1	1	373.243243
144304	211.0	1	1	247.303931

[ ]:
```
percentile_99 = df["Price"].quantile(0.99)
df_99 = df[df["Price"] <= percentile_99]

sns.displot(df_99, x="Price", color='#e3543b')
plt.title('Distribution of Prices')
plt.xlabel('Price')
plt.show()
```



[ ]:
```
df_99.sort_values(by='Price', ascending=False).head(10)
```

[ ]:

	Price	Body_type	Runned_Miles	Engin_size	Gear_box	Fuel_type	\
143611	82764.0	5	17500.0	3.8	0	9	
82763	82725.0	14	15.0	3.0	0	1	
144348	82500.0	5	22000.0	3.8	0	9	
82701	82500.0	14	10.0	4.4	0	1	
83392	82500.0	14	1250.0	4.4	0	1	
82990	82500.0	14	3063.0	3.0	0	1	
654	82500.0	4	18000.0	4.0	0	9	
518	82500.0	4	47000.0	6.0	0	9	
109190	82495.0	14	45000.0	5.5	0	9	
328	82450.0	5	17750.0	6.0	0	9	

	Engine_power	Wheelbase	Height	Width	Length	Average_mpg	\
143611	430.000000	2450.0	1295.0	1978.0	4509.0	32.500000	
82763	254.470000	2922.0	1869.0	2220.0	5000.0	40.900000	

```
144348    400.000000     2450.0  1303.0  1808.0  4491.0    30.700000
82701     334.000000     2922.0  1869.0  2220.0  5000.0    33.600000
83392     339.000000     2922.0  1835.0  2220.0  4999.0    33.600000
82990     254.470000     2922.0  1869.0  2220.0  5000.0    40.900000
654       521.000000     2746.0  1404.0  2227.0  4806.0    25.900000
518       563.295775     2746.0  1390.0  2226.0  4818.0    20.587021
109190    544.000000     2850.0  1938.0  1855.0  4763.0    20.500000
328       563.295775     2746.0  1391.0  2226.0  4818.0    20.587021

         Top_speed  Seat_num  Door_num  Gas_emission
143611  189.000000         1         1    247.303931
82763   130.000000         4         4    285.119658
144348  182.000000         3         1    247.303931
82701   135.000000         4         4    285.119658
83392   135.000000         4         4    285.119658
82990   130.000000         4         4    285.119658
654     191.000000         3         1    321.138528
518     196.853521         3         1    321.138528
109190  130.000000         4         4    309.836364
328     196.853521         3         1    321.138528
```

5.1.2 Outliers in the dataset: cleaning with the zscore

[ ]:
```
z_scores = zscore(df)
df = df[(abs(z_scores) < 4).all(axis=1)]
```

[ ]:
```
df.sort_values(by='Price', ascending=False).head(10)
```

[ ]:
```
          Price  Body_type  Runned_Miles  Engin_size  Gear_box  Fuel_type  \
143647  86990.0          5        6848.0         3.8         0          9
81962   86990.0         14         100.0         4.4         0          1
142877  86980.0          7        5545.0         4.0         0          1
83882   86980.0         14        1896.0         3.0         0          1
143680  86950.0          5       11500.0         3.8         0          9
14217   86895.0          4       14200.0         4.3         0          9
144760  86850.0          5        2639.0         3.0         0          9
84475   86844.0         14         894.0         3.0         0          1
144553  86800.0          5        6895.0         3.0         0          9
81991   86660.0         14        3552.0         3.0         0          1

        Engine_power  Wheelbase  Height   Width  Length  Average_mpg  \
143647        400.00     2450.0  1303.0  1808.0  4491.0         31.0
81962         334.00     2922.0  1869.0  2220.0  5000.0         33.6
142877        421.00     2950.0  1423.0  2165.0  5049.0         42.2
83882         301.72     2923.0  1811.0  2220.0  4879.0         40.4
143680        430.00     2450.0  1295.0  1978.0  4509.0         32.5
14217         460.00     2669.0  1308.0  1902.0  4562.0         21.5
```

```
144760        420.00     2450.0  1295.0  1978.0  4499.0         36.7
84475         301.72     2923.0  1811.0  2220.0  4879.0         40.4
144553        420.00     2450.0  1295.0  1978.0  4499.0         36.7
81991         254.47     2922.0  1869.0  2220.0  5000.0         40.9

        Top_speed  Seat_num  Door_num  Gas_emission
143647      184.0         3         1    247.303931
81962       135.0         4         4    285.119658
142877      177.0         3         4    206.956311
83882       140.0         4         4    241.497093
143680      189.0         1         1    247.303931
14217       193.0         3         1    288.642857
144760      190.0         3         1    247.303931
84475       140.0         4         4    241.497093
144553      190.0         3         1    247.303931
81991       130.0         4         4    285.119658
```

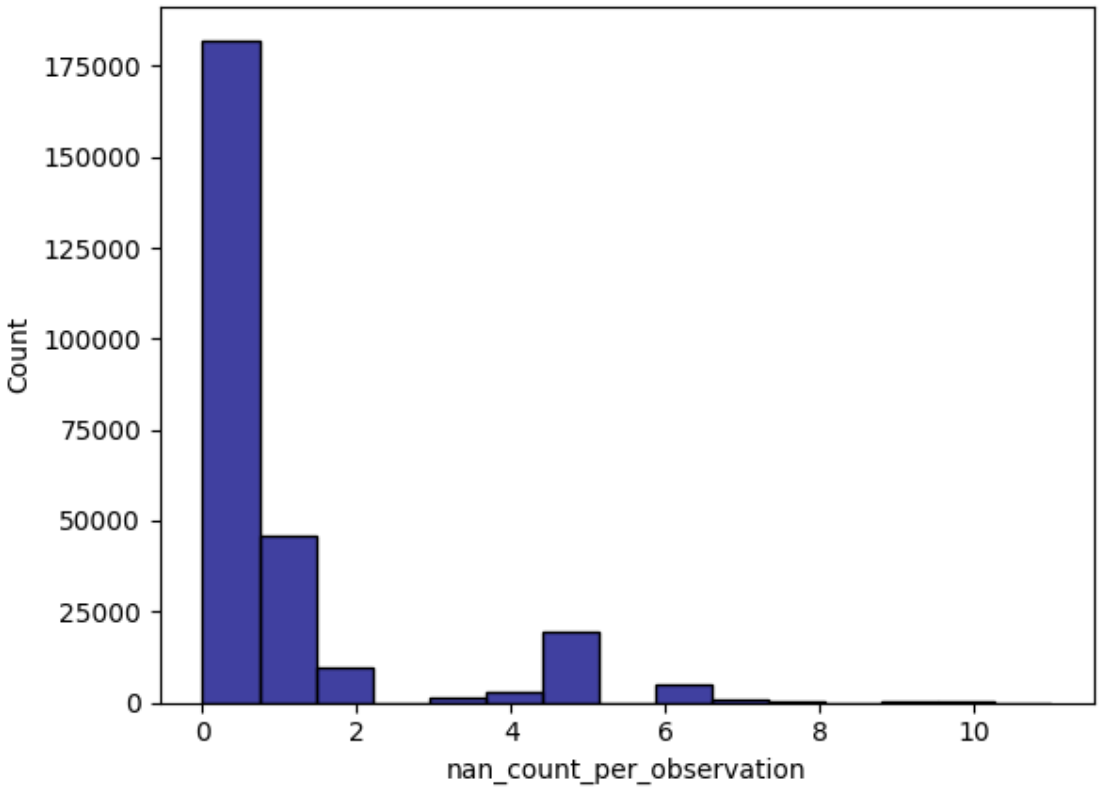
5.2 Analyse des NaN

[ ]:
```
df_na = preprocess_Ad_Table_Trim(Ad_table, Trim).get_full_data()
```

[ ]:
```
df_na['nan_count_per_observation'] = df_na.isnull().sum(axis=1)
```

[ ]:
```
sns.histplot(data=df_na, x="nan_count_per_observation", color="navy", bins=15)
```

[ ]:
```
<Axes: xlabel='nan_count_per_observation', ylabel='Count'>
```



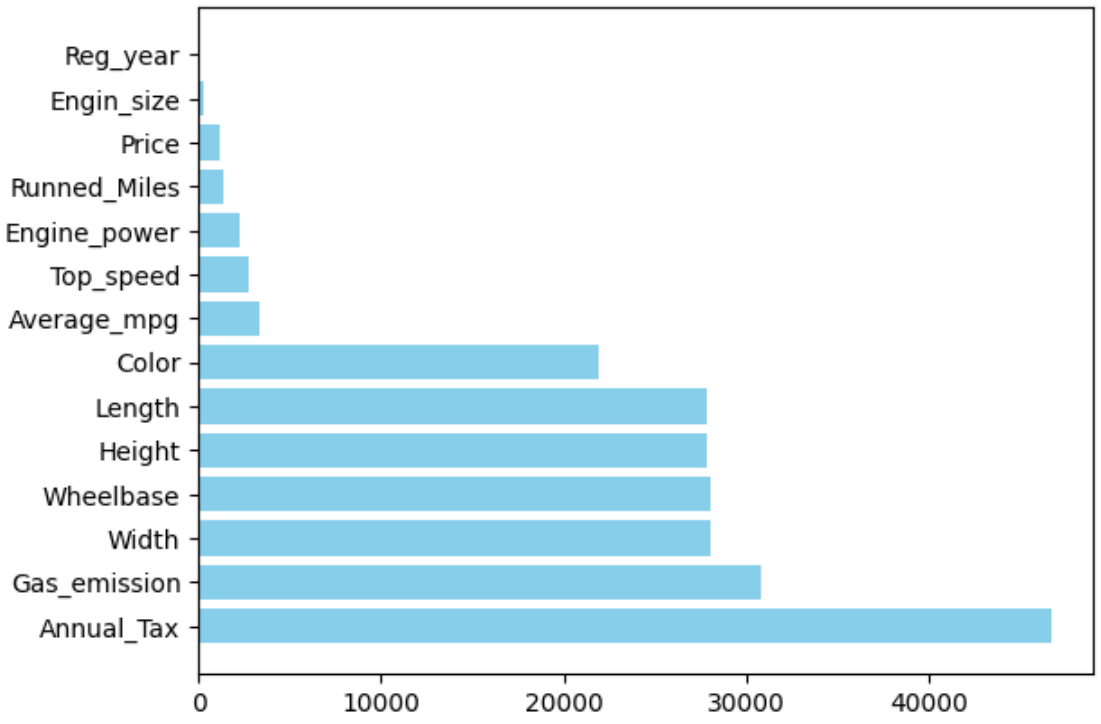
[ ]:
```
nan_count_per_variable = df_na.isnull().sum()
nan_count_per_variable
```

[ ]:
```
Maker                0
Genmodel             0
Genmodel_ID          0
Adv_ID               0
Adv_year             0
Adv_month            0
Color            21875
Reg_year             7
Body_type            0
Runned_Miles      1313
Engin_size         323
Gear_box             0
Fuel_type            0
Price             1145
Engine_power      2230
Annual_Tax       46675
Wheelbase        27998
Height           27801
```

```
Width                          28080
Length                         27801
Average_mpg                     3307
Top_speed                       2758
Seat_num                           0
Door_num                           0
Gas_emission                   30786
nan_count_per_observation          0
dtype: int64
```

[ ]:
```
nan_count_table = pd.DataFrame({'Variable': nan_count_per_variable.index, 'NaN␣
 ↪Count': nan_count_per_variable.values})
nan_count_table = nan_count_table[nan_count_table['NaN Count']>0].
 ↪sort_values(by='NaN Count', ascending=False)

plt.barh(nan_count_table['Variable'], nan_count_table['NaN Count'],␣
 ↪color='skyblue')
```

[ ]:
```
<BarContainer object of 14 artists>
```



We decided to delete the columns : Annual_Tax and Color because it was with lot of NaN and it doesn't really have an impact on the price

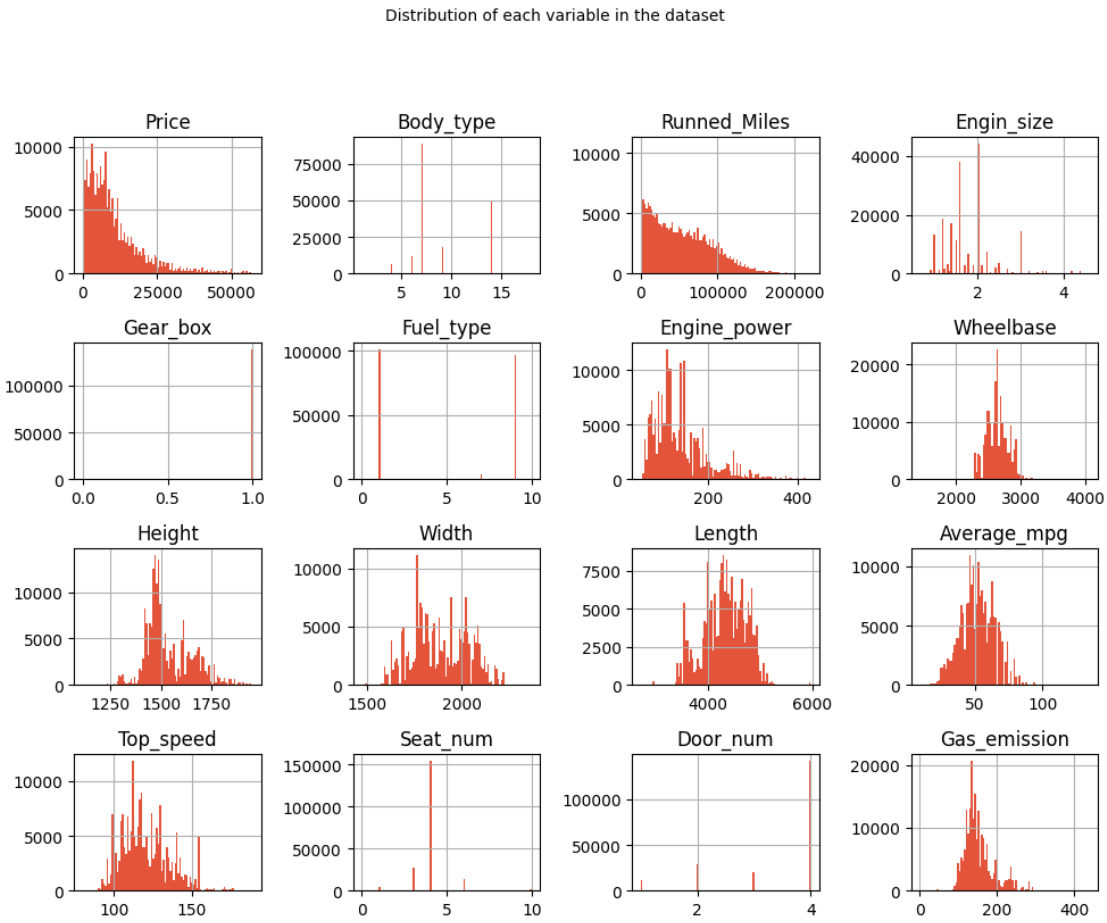
5.3 Distribution

5.3.1 Distribution of each variable

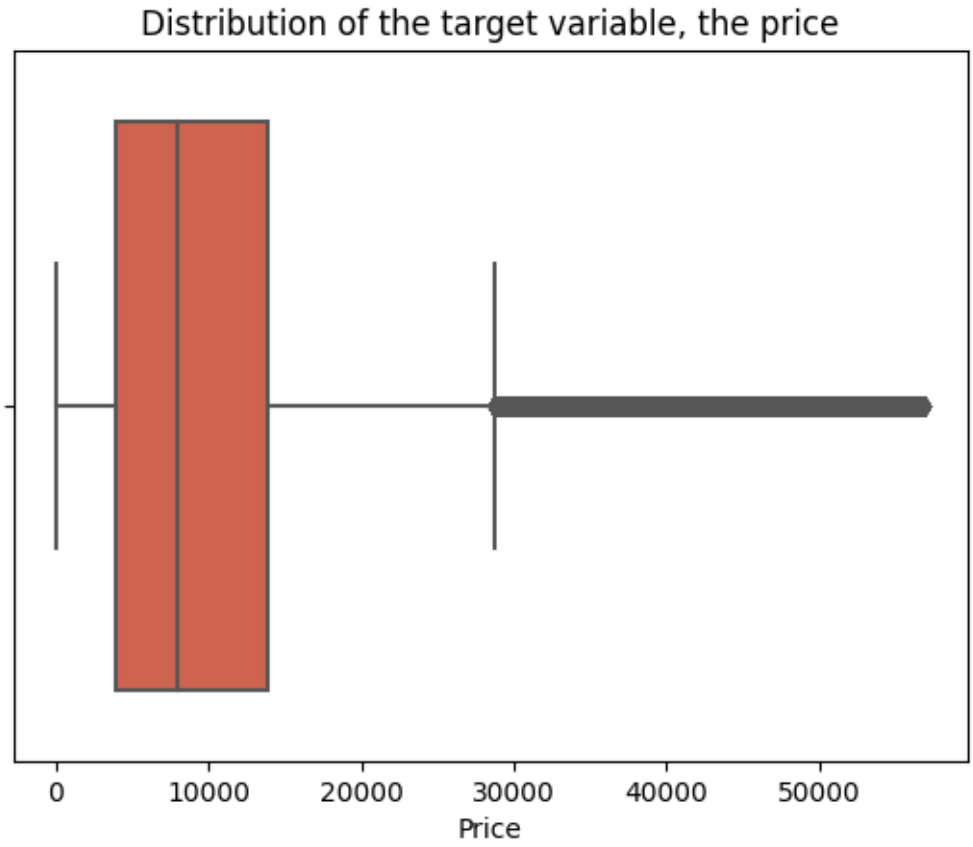
We now analyse the final dataset for the graphs

[ ]:
```
y, X = preprocess_Ad_Table_Trim(Ad_table, Trim).final_set(
                one_hot_encoder = False,
                standardization = False)

df = pd.concat([y, X], axis=1)
```

[ ]:
```
df.hist(bins=100, figsize=(10, 8), color='#e3543b')
plt.suptitle('Distribution of each variable in the dataset', x=0.5, y=1.02,␣
 ↪fontsize=10)
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



[ ]:
```
sns.boxplot(x='Price', data=df, color='#e3543b')
plt.title('Distribution of the target variable, the price')
plt.xlabel('Price')
plt.show()
```

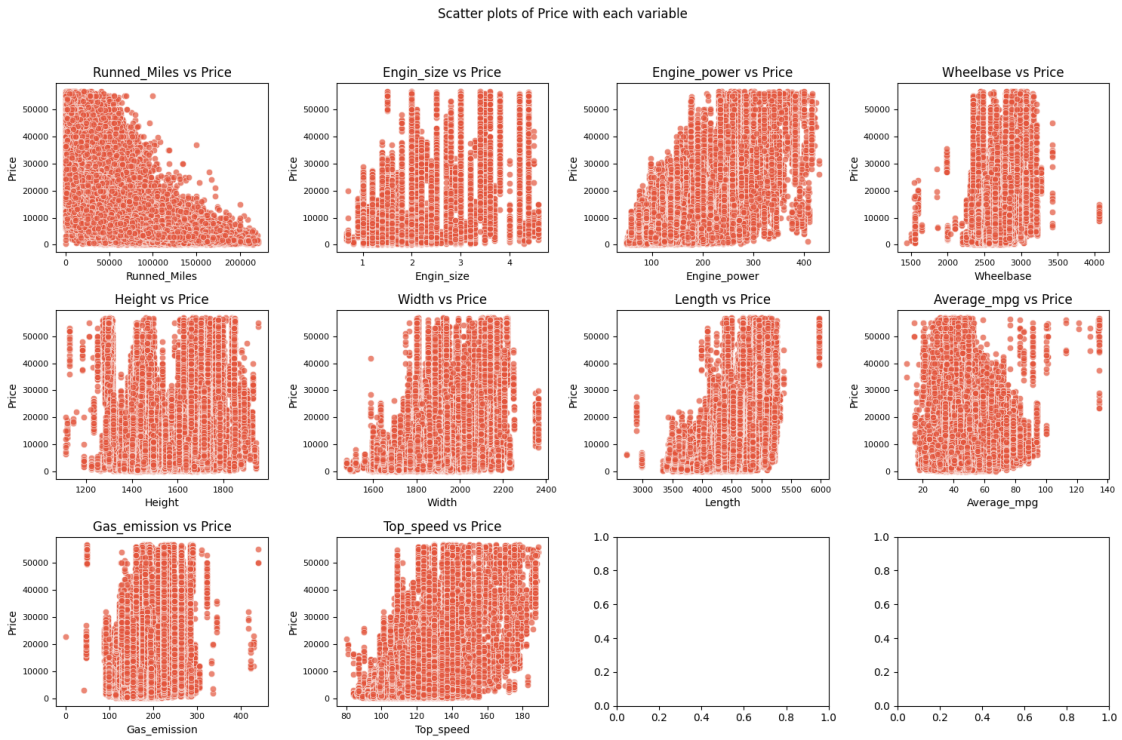


5.3.2 Relationship between the dependant variable and the independent variables

Float variables

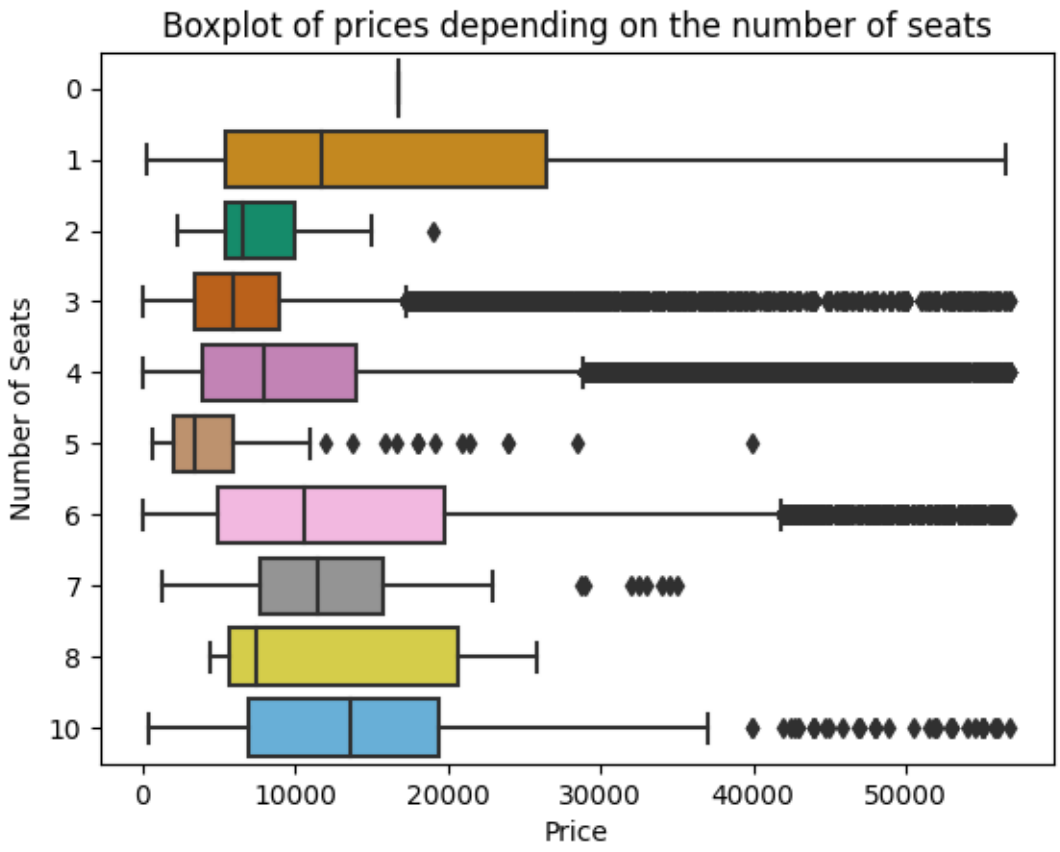
[ ]:
```
variables_numeriques = ['Runned_Miles', 'Engin_size', 'Engine_power',␣
↪'Wheelbase', 'Height', 'Width', 'Length', 'Average_mpg', 'Gas_emission',␣
↪'Top_speed']

fig, axes = plt.subplots(nrows=3, ncols=4, figsize=(15, 10))
fig.suptitle('Scatter plots of Price with each variable')

for i, variable in enumerate(variables_numeriques):
    if i < 10:
        row = i // 4
        col = i % 4
```

```
        sns.scatterplot(x=variable, y='Price', data=df, ax=axes[row, col],␣
↪color='#e3543b', alpha=0.7)
        axes[row, col].set_title(f'{variable} vs Price', fontsize=12)
        axes[row, col].set_xlabel(variable, fontsize=10)
        axes[row, col].set_ylabel('Price', fontsize=10)
        axes[row, col].tick_params(axis='both', which='both', labelsize=8)
    else:
        fig.delaxes(axes.flatten()[i])

plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

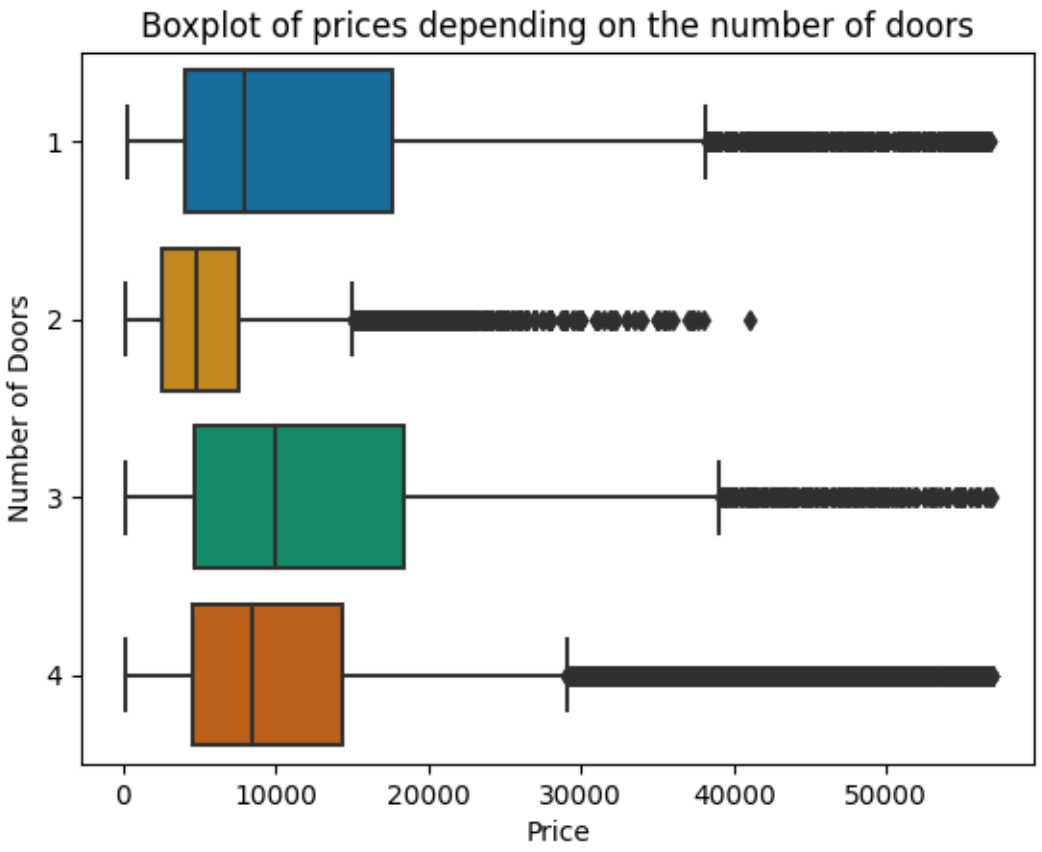


Integers variables

[ ]:
```
sns.boxplot(x='Price', y='Seat_num', data=df, orient='h', palette="colorblind")
plt.title('Boxplot of prices depending on the number of seats')
plt.xlabel('Price')
plt.ylabel('Number of Seats')
plt.show()
```

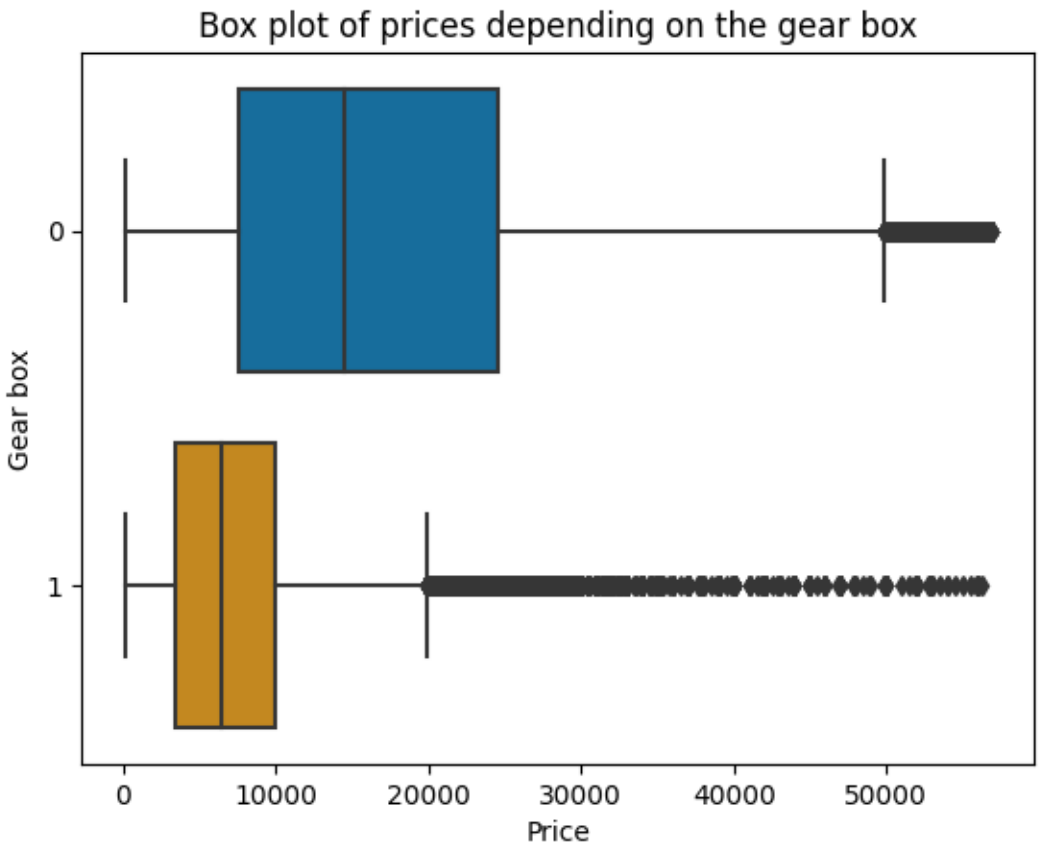


[ ]:
```
sns.boxplot(x='Price', y='Door_num', data=df, orient='h', palette="colorblind")
plt.title('Boxplot of prices depending on the number of doors')
plt.xlabel('Price')
plt.ylabel('Number of Doors')
plt.show()
```

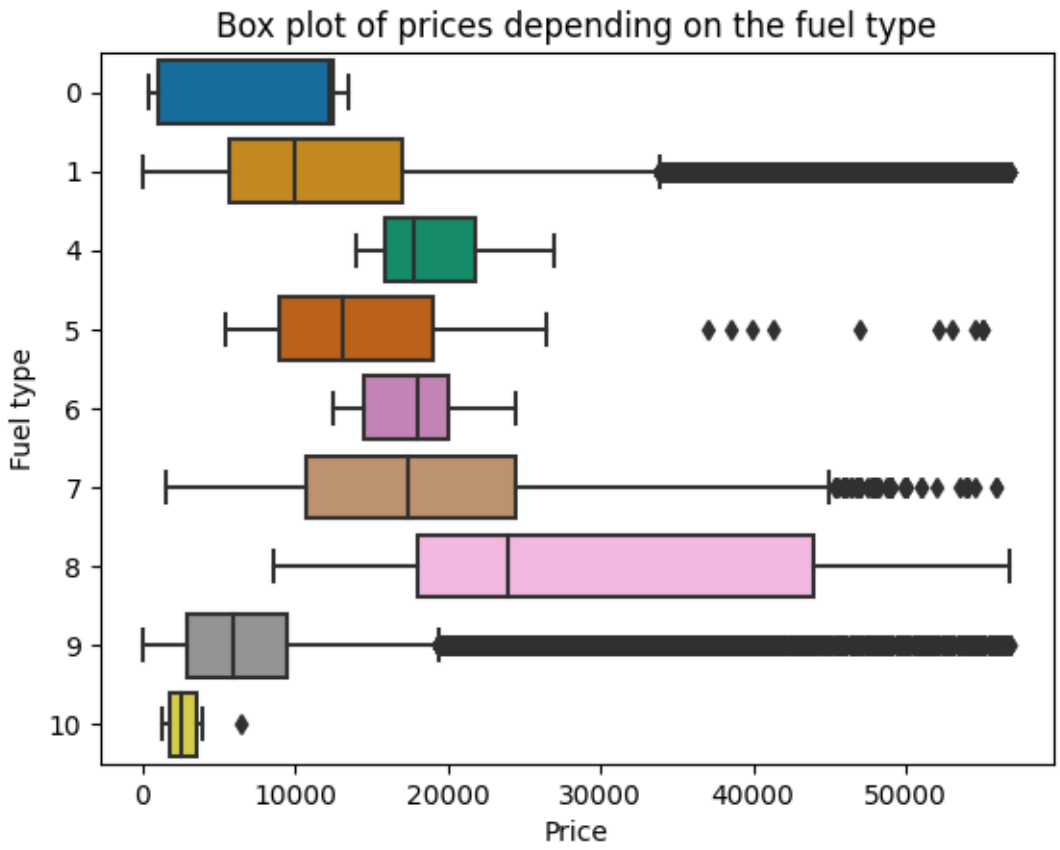


Qualitative variables

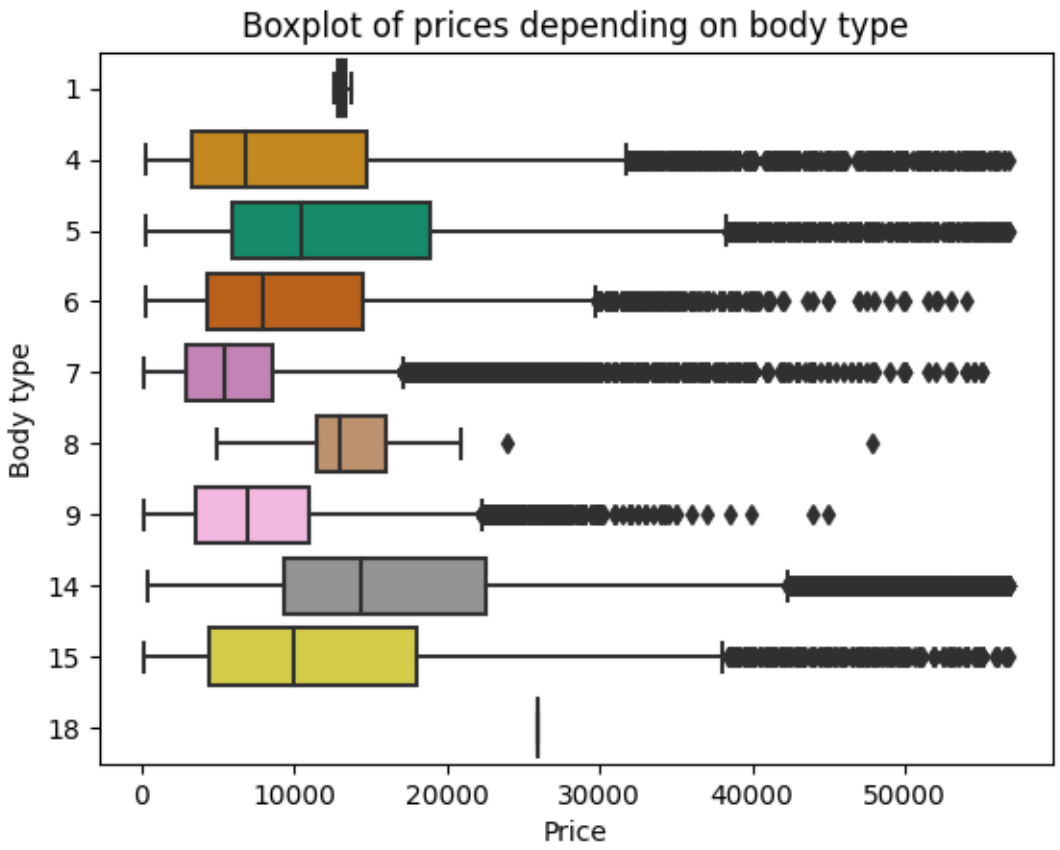
[ ]:
```
sns.boxplot(x='Price', y='Gear_box', data=df, orient='h', palette="colorblind")
plt.title('Box plot of prices depending on the gear box')
plt.ylabel('Gear box')
plt.xlabel('Price')
plt.show()
```



[ ]:
```
sns.boxplot(x='Price', y='Fuel_type', data=df, orient='h', palette="colorblind")
plt.title('Box plot of prices depending on the fuel type')
plt.ylabel('Fuel type')
plt.xlabel('Price')
plt.show()
```



[ ]:
```
sns.boxplot(x='Price', y='Body_type', data=df, orient='h', palette="colorblind")
plt.title('Boxplot of prices depending on body type')
plt.xlabel('Price')
plt.ylabel('Body type')
plt.show()
```



5.4 Correlation matrix

[ ]:
```
y, X = preprocess_Ad_Table_Trim(Ad_table, Trim).final_set(
                standardization=True,
                remove_outliers=True,
                columns_to_drop=columns_to_drop)
```

[ ]:
```
correlation_matrix = X.corr()

plt.figure(figsize=(25, 10))
heatmap = sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.
 ↪2f')
plt.title('Correlation matrix')
heatmap.set_xticklabels(heatmap.get_xticklabels(), rotation=90)
heatmap.set_yticklabels(heatmap.get_yticklabels(), rotation=0)
plt.show()
```

6 Machine learning models

6.1 Classical data

6.1.1 XGBoost

Setting Data

[ ]:
```
# Read data
Ad_table = pd.read_csv('/content/drive/MyDrive/ML BigProject/data/Ad_table␣
 ↪(extra).csv')
Trim = pd.read_csv('/content/drive/MyDrive/ML BigProject/data/Trim_table.csv')
preprocess_data = preprocess_Ad_Table_Trim(Ad_table, Trim)

# Apply self-made function and obtain cleaned data
y, X = preprocess_data.final_set(remove_outliers=True, standardization=False,␣
 ↪columns_to_drop = columns_to_drop)

# Create numpy arrays
X_values = X.values
y_values = y.values
```

[ ]:
```
# Train-Test-Split
X_train, X_test, y_train, y_test = train_test_split(X_values, y_values,␣
 ↪random_state = RANDOM_STATE, test_size = 0.15)

# Train-Validation-Split for the initial XGBoost Model
X_train_init, X_val, y_train_init, y_val = train_test_split(X_train, y_train,␣
 ↪random_state = RANDOM_STATE, test_size = 0.15)

# 72% training data
# 12.75% validation data
# 15% test data
```

Model Training

A) Train Baseline XGBoost Model

[ ]:
```
# Define initial XGB model to obtain optimal number of estimators (trees) for a␣
 ↪given learning rate eta
xgb_init = xgb.XGBRegressor(objective = "reg:squarederror", # Objective␣
 ↪function to be minimized by XGBoost model
                            eval_metric="rmse",
                            base_score = 0.5, # Initial prediction score for␣
 ↪all observations (default value).
                            booster = "gbtree", # Boosting type, here gradient␣
 ↪boosting tree
                            learning_rate = 0.1, # Learning rate of XGBoost␣
 ↪model determining the speed of convergence (default value)
```

```
                            n_estimators = 5000, # Number of trees to be␣
 ↪estimated. Here, set to very high (unreached) number because stopped by␣
 ↪early_stopping_rounds.
                            max_depth = 8, # Maximum tree depth (default value␣
 ↪= 6). Since we have p=14>6, we allow tree depth up to 8 levels. To be tuned␣
 ↪later.
                            min_child_weight = 1, # Minimum number of instances␣
 ↪needed per child (default value). To be tunded later.
                            reg_lambda = 1, # L2 regularization parameter␣
 ↪scaling the similiratiy scores and controlling for nodes containing only few␣
 ↪parameters. To be tuned later.
                            gamma = 0.1, # Similiarity score threshold (pruning␣
 ↪parameter) to create new split. To be tuned later.
                            early_stopping_rounds = 500, # Number of iterations␣
 ↪in which validation metric needs to improve at least once to continue model␣
 ↪training (10% of total iterations)
                            subsample = 0.8, # Sample size of training data to␣
 ↪be used at each iteration. To be tuned, later.
                            sampling_method = "uniform", # Determines␣
 ↪distribution from which subsample is randomly drawn.
                            importance_type = "gain", # Criteria for the␣
 ↪model's feature_importance property
                            n_jobs = 7, # Number of CPU kernels to be used for␣
 ↪model training (time reduction)
                            verbosity = 2,
                            random_state = 999
                            )

# Fit model and obtain RMSE using cross-validation (eval_set)
xgb_init.fit(X_train_init, y_train_init, eval_set = [(X_train_init,␣
 ↪y_train_init), (X_val, y_val)])
```

```
[0]     validation_0-rmse:12893.99379   validation_1-rmse:12873.89577
[1]     validation_0-rmse:11681.02078   validation_1-rmse:11664.64635
[2]     validation_0-rmse:10595.21346   validation_1-rmse:10581.98545
[3]     validation_0-rmse:9617.30602    validation_1-rmse:9605.79690
[4]     validation_0-rmse:8740.84876    validation_1-rmse:8734.62190
[5]     validation_0-rmse:7956.14337    validation_1-rmse:7956.79002
[6]     validation_0-rmse:7250.68804    validation_1-rmse:7257.25042
[7]     validation_0-rmse:6618.28265    validation_1-rmse:6628.57434
[8]     validation_0-rmse:6053.00069    validation_1-rmse:6068.67158
[9]     validation_0-rmse:5546.88014    validation_1-rmse:5566.10598
[10]    validation_0-rmse:5095.76418    validation_1-rmse:5119.92900
[11]    validation_0-rmse:4693.80816    validation_1-rmse:4720.94706
[12]    validation_0-rmse:4337.74018    validation_1-rmse:4366.59049
[13]    validation_0-rmse:4020.44318    validation_1-rmse:4052.71267
[14]    validation_0-rmse:3735.00596    validation_1-rmse:3769.38890
```

```
[15]    validation_0-rmse:3487.08324    validation_1-rmse:3526.82998
[16]    validation_0-rmse:3268.06626    validation_1-rmse:3313.24492
[17]    validation_0-rmse:3078.83176    validation_1-rmse:3129.17039
[18]    validation_0-rmse:2904.68687    validation_1-rmse:2959.63647
[19]    validation_0-rmse:2759.20283    validation_1-rmse:2818.63452
[20]    validation_0-rmse:2629.54638    validation_1-rmse:2693.33060
[21]    validation_0-rmse:2513.96986    validation_1-rmse:2581.28480
[22]    validation_0-rmse:2418.88874    validation_1-rmse:2489.86106
[23]    validation_0-rmse:2334.90135    validation_1-rmse:2409.63566
[24]    validation_0-rmse:2254.44867    validation_1-rmse:2333.07047
[25]    validation_0-rmse:2188.94930    validation_1-rmse:2271.74716
[26]    validation_0-rmse:2127.67639    validation_1-rmse:2215.05622
[27]    validation_0-rmse:2077.51829    validation_1-rmse:2169.16512
[28]    validation_0-rmse:2030.48051    validation_1-rmse:2126.27665
[29]    validation_0-rmse:1995.27213    validation_1-rmse:2093.79795
[30]    validation_0-rmse:1961.69582    validation_1-rmse:2062.68249
[31]    validation_0-rmse:1934.27842    validation_1-rmse:2037.47278
[32]    validation_0-rmse:1905.70750    validation_1-rmse:2011.11449
[33]    validation_0-rmse:1882.27457    validation_1-rmse:1989.96651
[34]    validation_0-rmse:1861.39263    validation_1-rmse:1971.80708
[35]    validation_0-rmse:1842.31527    validation_1-rmse:1953.58486
[36]    validation_0-rmse:1824.77122    validation_1-rmse:1937.05438
[37]    validation_0-rmse:1803.69766    validation_1-rmse:1918.21583
[38]    validation_0-rmse:1789.16081    validation_1-rmse:1905.35313
[39]    validation_0-rmse:1771.95144    validation_1-rmse:1890.01698
[40]    validation_0-rmse:1760.36508    validation_1-rmse:1880.34470
[41]    validation_0-rmse:1748.47433    validation_1-rmse:1869.01719
[42]    validation_0-rmse:1737.15949    validation_1-rmse:1859.98692
[43]    validation_0-rmse:1725.45740    validation_1-rmse:1849.38768
[44]    validation_0-rmse:1720.19265    validation_1-rmse:1844.46154
[45]    validation_0-rmse:1709.70612    validation_1-rmse:1834.86858
[46]    validation_0-rmse:1700.20261    validation_1-rmse:1827.22997
[47]    validation_0-rmse:1694.43098    validation_1-rmse:1822.34478
[48]    validation_0-rmse:1688.81376    validation_1-rmse:1818.00088
[49]    validation_0-rmse:1680.61538    validation_1-rmse:1811.22094
[50]    validation_0-rmse:1671.24671    validation_1-rmse:1803.34649
[51]    validation_0-rmse:1666.95522    validation_1-rmse:1800.17943
[52]    validation_0-rmse:1662.85130    validation_1-rmse:1796.79294
[53]    validation_0-rmse:1659.67229    validation_1-rmse:1793.86036
[54]    validation_0-rmse:1655.93903    validation_1-rmse:1791.17741
[55]    validation_0-rmse:1651.27760    validation_1-rmse:1787.45308
[56]    validation_0-rmse:1647.05432    validation_1-rmse:1783.85078
[57]    validation_0-rmse:1641.96804    validation_1-rmse:1779.46113
[58]    validation_0-rmse:1637.28168    validation_1-rmse:1775.61343
[59]    validation_0-rmse:1632.35085    validation_1-rmse:1771.65286
[60]    validation_0-rmse:1629.50201    validation_1-rmse:1769.56444
[61]    validation_0-rmse:1627.14749    validation_1-rmse:1767.81941
[62]    validation_0-rmse:1621.98240    validation_1-rmse:1763.47633
```

```
[63]    validation_0-rmse:1617.35688    validation_1-rmse:1760.08702
[64]    validation_0-rmse:1612.48297    validation_1-rmse:1756.44574
[65]    validation_0-rmse:1607.18014    validation_1-rmse:1752.55542
[66]    validation_0-rmse:1602.37069    validation_1-rmse:1749.51707
[67]    validation_0-rmse:1597.29206    validation_1-rmse:1745.85207
[68]    validation_0-rmse:1593.05341    validation_1-rmse:1742.58271
[69]    validation_0-rmse:1589.40823    validation_1-rmse:1740.16805
[70]    validation_0-rmse:1584.93809    validation_1-rmse:1736.30340
[71]    validation_0-rmse:1580.37890    validation_1-rmse:1732.31075
[72]    validation_0-rmse:1577.87627    validation_1-rmse:1730.60582
[73]    validation_0-rmse:1573.95267    validation_1-rmse:1727.72274
[74]    validation_0-rmse:1571.47071    validation_1-rmse:1725.70818
[75]    validation_0-rmse:1566.09788    validation_1-rmse:1721.85497
[76]    validation_0-rmse:1563.56930    validation_1-rmse:1720.19062
[77]    validation_0-rmse:1558.62302    validation_1-rmse:1716.48604
[78]    validation_0-rmse:1554.40356    validation_1-rmse:1712.75602
[79]    validation_0-rmse:1551.46221    validation_1-rmse:1710.61084
[80]    validation_0-rmse:1547.79957    validation_1-rmse:1707.30904
[81]    validation_0-rmse:1544.70159    validation_1-rmse:1704.52423
[82]    validation_0-rmse:1541.23696    validation_1-rmse:1702.26947
[83]    validation_0-rmse:1537.79286    validation_1-rmse:1699.48276
[84]    validation_0-rmse:1534.16025    validation_1-rmse:1696.87926
[85]    validation_0-rmse:1529.80370    validation_1-rmse:1693.90048
[86]    validation_0-rmse:1525.87250    validation_1-rmse:1690.78702
[87]    validation_0-rmse:1523.99307    validation_1-rmse:1689.74042
[88]    validation_0-rmse:1520.21761    validation_1-rmse:1686.94713
[89]    validation_0-rmse:1519.25689    validation_1-rmse:1686.13022
[90]    validation_0-rmse:1517.02714    validation_1-rmse:1684.55366
[91]    validation_0-rmse:1514.74135    validation_1-rmse:1683.22310
[92]    validation_0-rmse:1512.88119    validation_1-rmse:1681.90968
[93]    validation_0-rmse:1510.49898    validation_1-rmse:1680.31194
[94]    validation_0-rmse:1507.44963    validation_1-rmse:1677.49371
[95]    validation_0-rmse:1505.40173    validation_1-rmse:1676.14346
[96]    validation_0-rmse:1503.17493    validation_1-rmse:1674.26651
[97]    validation_0-rmse:1499.75601    validation_1-rmse:1670.88649
[98]    validation_0-rmse:1497.29967    validation_1-rmse:1669.14708
[99]    validation_0-rmse:1495.16747    validation_1-rmse:1667.61046
[100]   validation_0-rmse:1490.79349    validation_1-rmse:1663.87841
[101]   validation_0-rmse:1487.44586    validation_1-rmse:1661.34417
[102]   validation_0-rmse:1485.50152    validation_1-rmse:1660.28980
[103]   validation_0-rmse:1482.69052    validation_1-rmse:1658.09492
[104]   validation_0-rmse:1480.70598    validation_1-rmse:1657.16408
[105]   validation_0-rmse:1478.19698    validation_1-rmse:1655.04013
[106]   validation_0-rmse:1474.87552    validation_1-rmse:1652.57139
[107]   validation_0-rmse:1472.28082    validation_1-rmse:1650.58917
[108]   validation_0-rmse:1470.27170    validation_1-rmse:1648.88260
[109]   validation_0-rmse:1468.95468    validation_1-rmse:1647.90221
[110]   validation_0-rmse:1467.04632    validation_1-rmse:1646.49534
```

```
[111]   validation_0-rmse:1464.04504    validation_1-rmse:1644.52152
[112]   validation_0-rmse:1462.07355    validation_1-rmse:1643.17385
[113]   validation_0-rmse:1459.96684    validation_1-rmse:1641.94934
[114]   validation_0-rmse:1457.86104    validation_1-rmse:1640.49817
[115]   validation_0-rmse:1455.26031    validation_1-rmse:1638.65816
[116]   validation_0-rmse:1453.61971    validation_1-rmse:1637.58360
[117]   validation_0-rmse:1451.41179    validation_1-rmse:1636.28122
[118]   validation_0-rmse:1449.92731    validation_1-rmse:1635.65319
[119]   validation_0-rmse:1449.06584    validation_1-rmse:1635.08689
[120]   validation_0-rmse:1447.97786    validation_1-rmse:1634.27012
[121]   validation_0-rmse:1446.33736    validation_1-rmse:1632.99436
[122]   validation_0-rmse:1444.63738    validation_1-rmse:1631.67514
[123]   validation_0-rmse:1443.13895    validation_1-rmse:1630.81063
[124]   validation_0-rmse:1440.37113    validation_1-rmse:1628.83216
[125]   validation_0-rmse:1439.22120    validation_1-rmse:1627.68004
[126]   validation_0-rmse:1437.46076    validation_1-rmse:1626.77016
[127]   validation_0-rmse:1435.86011    validation_1-rmse:1626.10033
[128]   validation_0-rmse:1434.49417    validation_1-rmse:1624.98020
[129]   validation_0-rmse:1433.10562    validation_1-rmse:1624.50490
[130]   validation_0-rmse:1430.58423    validation_1-rmse:1621.98225
[131]   validation_0-rmse:1428.78175    validation_1-rmse:1620.48574
[132]   validation_0-rmse:1427.17428    validation_1-rmse:1618.94675
[133]   validation_0-rmse:1424.62463    validation_1-rmse:1617.07393
[134]   validation_0-rmse:1422.60019    validation_1-rmse:1616.19210
[135]   validation_0-rmse:1420.66862    validation_1-rmse:1614.73588
[136]   validation_0-rmse:1419.86744    validation_1-rmse:1614.77195
[137]   validation_0-rmse:1418.47875    validation_1-rmse:1613.64062
[138]   validation_0-rmse:1416.94024    validation_1-rmse:1612.36739
[139]   validation_0-rmse:1414.70733    validation_1-rmse:1610.82025
[140]   validation_0-rmse:1412.84703    validation_1-rmse:1609.65813
[141]   validation_0-rmse:1410.56539    validation_1-rmse:1608.61673
[142]   validation_0-rmse:1408.87237    validation_1-rmse:1607.41156
[143]   validation_0-rmse:1407.29910    validation_1-rmse:1606.31766
[144]   validation_0-rmse:1405.54361    validation_1-rmse:1605.10797
[145]   validation_0-rmse:1403.87936    validation_1-rmse:1603.85976
[146]   validation_0-rmse:1402.09958    validation_1-rmse:1602.33210
[147]   validation_0-rmse:1400.88458    validation_1-rmse:1601.59524
[148]   validation_0-rmse:1399.71593    validation_1-rmse:1600.83492
[149]   validation_0-rmse:1398.17722    validation_1-rmse:1599.87586
[150]   validation_0-rmse:1396.41995    validation_1-rmse:1598.77878
[151]   validation_0-rmse:1395.60574    validation_1-rmse:1598.36675
[152]   validation_0-rmse:1393.91610    validation_1-rmse:1597.21780
[153]   validation_0-rmse:1392.00655    validation_1-rmse:1595.98650
[154]   validation_0-rmse:1390.24564    validation_1-rmse:1594.95942
[155]   validation_0-rmse:1389.27747    validation_1-rmse:1594.35899
[156]   validation_0-rmse:1388.05143    validation_1-rmse:1593.67247
[157]   validation_0-rmse:1385.89060    validation_1-rmse:1592.53015
[158]   validation_0-rmse:1384.27035    validation_1-rmse:1591.30146
```

```
[159]   validation_0-rmse:1382.82549    validation_1-rmse:1590.58474
[160]   validation_0-rmse:1380.88665    validation_1-rmse:1589.36011
[161]   validation_0-rmse:1379.52144    validation_1-rmse:1588.55778
[162]   validation_0-rmse:1378.13782    validation_1-rmse:1588.24042
[163]   validation_0-rmse:1376.44942    validation_1-rmse:1587.42904
[164]   validation_0-rmse:1375.66588    validation_1-rmse:1586.92347
[165]   validation_0-rmse:1374.62974    validation_1-rmse:1586.53882
[166]   validation_0-rmse:1373.52744    validation_1-rmse:1585.96031
[167]   validation_0-rmse:1372.12930    validation_1-rmse:1585.69841
[168]   validation_0-rmse:1371.17994    validation_1-rmse:1585.15714
[169]   validation_0-rmse:1370.03441    validation_1-rmse:1584.21846
[170]   validation_0-rmse:1369.16579    validation_1-rmse:1583.86070
[171]   validation_0-rmse:1368.36883    validation_1-rmse:1583.38025
[172]   validation_0-rmse:1367.08016    validation_1-rmse:1582.90429
[173]   validation_0-rmse:1365.99148    validation_1-rmse:1582.62489
[174]   validation_0-rmse:1364.55288    validation_1-rmse:1581.54456
[175]   validation_0-rmse:1363.54329    validation_1-rmse:1581.56853
[176]   validation_0-rmse:1361.75228    validation_1-rmse:1580.49001
[177]   validation_0-rmse:1360.61017    validation_1-rmse:1579.78774
[178]   validation_0-rmse:1360.34855    validation_1-rmse:1579.74088
[179]   validation_0-rmse:1359.25929    validation_1-rmse:1579.05682
[180]   validation_0-rmse:1358.13564    validation_1-rmse:1578.28899
[181]   validation_0-rmse:1356.74795    validation_1-rmse:1577.49060
[182]   validation_0-rmse:1355.43245    validation_1-rmse:1576.78226
[183]   validation_0-rmse:1353.81590    validation_1-rmse:1575.70948
[184]   validation_0-rmse:1352.26951    validation_1-rmse:1574.81013
[185]   validation_0-rmse:1350.94623    validation_1-rmse:1574.13196
[186]   validation_0-rmse:1349.85415    validation_1-rmse:1573.54435
[187]   validation_0-rmse:1347.88655    validation_1-rmse:1572.43317
[188]   validation_0-rmse:1346.30677    validation_1-rmse:1571.42703
[189]   validation_0-rmse:1344.94365    validation_1-rmse:1570.76397
[190]   validation_0-rmse:1343.93589    validation_1-rmse:1570.03602
[191]   validation_0-rmse:1342.87893    validation_1-rmse:1569.66867
[192]   validation_0-rmse:1342.25183    validation_1-rmse:1569.43259
[193]   validation_0-rmse:1341.38892    validation_1-rmse:1569.44579
[194]   validation_0-rmse:1340.97171    validation_1-rmse:1569.16068
[195]   validation_0-rmse:1340.02006    validation_1-rmse:1568.98145
[196]   validation_0-rmse:1338.80632    validation_1-rmse:1568.25822
[197]   validation_0-rmse:1337.39025    validation_1-rmse:1567.65919
[198]   validation_0-rmse:1335.62227    validation_1-rmse:1566.54138
[199]   validation_0-rmse:1334.41688    validation_1-rmse:1566.18214
[200]   validation_0-rmse:1333.20929    validation_1-rmse:1565.78435
[201]   validation_0-rmse:1331.62161    validation_1-rmse:1565.35240
[202]   validation_0-rmse:1330.42290    validation_1-rmse:1564.81071
[203]   validation_0-rmse:1329.04031    validation_1-rmse:1564.06111
[204]   validation_0-rmse:1327.81180    validation_1-rmse:1563.97366
[205]   validation_0-rmse:1326.81130    validation_1-rmse:1563.16604
[206]   validation_0-rmse:1325.09295    validation_1-rmse:1562.35401
```

```
[207]   validation_0-rmse:1323.95832    validation_1-rmse:1561.75830
[208]   validation_0-rmse:1322.35430    validation_1-rmse:1560.72070
[209]   validation_0-rmse:1321.40346    validation_1-rmse:1560.37624
[210]   validation_0-rmse:1320.66504    validation_1-rmse:1559.88350
[211]   validation_0-rmse:1319.67249    validation_1-rmse:1559.42496
[212]   validation_0-rmse:1318.52113    validation_1-rmse:1558.88986
[213]   validation_0-rmse:1317.84933    validation_1-rmse:1558.33266
[214]   validation_0-rmse:1316.70171    validation_1-rmse:1557.62793
[215]   validation_0-rmse:1315.66863    validation_1-rmse:1556.79871
[216]   validation_0-rmse:1314.69848    validation_1-rmse:1556.51549
[217]   validation_0-rmse:1313.54940    validation_1-rmse:1556.04234
[218]   validation_0-rmse:1312.68275    validation_1-rmse:1555.61744
[219]   validation_0-rmse:1312.14000    validation_1-rmse:1555.39815
[220]   validation_0-rmse:1311.09359    validation_1-rmse:1554.87353
[221]   validation_0-rmse:1310.53265    validation_1-rmse:1554.66387
[222]   validation_0-rmse:1309.86227    validation_1-rmse:1554.29376
[223]   validation_0-rmse:1308.86711    validation_1-rmse:1553.59939
[224]   validation_0-rmse:1307.28891    validation_1-rmse:1552.47009
[225]   validation_0-rmse:1306.26000    validation_1-rmse:1552.40728
[226]   validation_0-rmse:1305.14779    validation_1-rmse:1551.99601
[227]   validation_0-rmse:1304.13915    validation_1-rmse:1552.02230
[228]   validation_0-rmse:1303.00356    validation_1-rmse:1551.60663
[229]   validation_0-rmse:1302.43397    validation_1-rmse:1551.58354
[230]   validation_0-rmse:1301.63798    validation_1-rmse:1551.32335
[231]   validation_0-rmse:1300.41993    validation_1-rmse:1550.85752
[232]   validation_0-rmse:1299.64966    validation_1-rmse:1550.65423
[233]   validation_0-rmse:1299.44515    validation_1-rmse:1550.67562
[234]   validation_0-rmse:1298.93140    validation_1-rmse:1550.74085
[235]   validation_0-rmse:1297.34559    validation_1-rmse:1549.64566
[236]   validation_0-rmse:1296.68173    validation_1-rmse:1549.39524
[237]   validation_0-rmse:1296.09825    validation_1-rmse:1549.16195
[238]   validation_0-rmse:1295.49272    validation_1-rmse:1548.94955
[239]   validation_0-rmse:1294.57101    validation_1-rmse:1548.76174
[240]   validation_0-rmse:1293.60657    validation_1-rmse:1548.10011
[241]   validation_0-rmse:1292.42160    validation_1-rmse:1547.50623
[242]   validation_0-rmse:1291.27852    validation_1-rmse:1546.58538
[243]   validation_0-rmse:1290.15075    validation_1-rmse:1545.89233
[244]   validation_0-rmse:1289.15003    validation_1-rmse:1545.39144
[245]   validation_0-rmse:1288.05089    validation_1-rmse:1544.90246
[246]   validation_0-rmse:1287.28146    validation_1-rmse:1544.37892
[247]   validation_0-rmse:1286.34806    validation_1-rmse:1544.12336
[248]   validation_0-rmse:1285.77476    validation_1-rmse:1543.90218
[249]   validation_0-rmse:1284.67091    validation_1-rmse:1543.55941
[250]   validation_0-rmse:1283.98915    validation_1-rmse:1543.50470
[251]   validation_0-rmse:1282.68259    validation_1-rmse:1542.81108
[252]   validation_0-rmse:1281.88958    validation_1-rmse:1542.58173
[253]   validation_0-rmse:1280.57380    validation_1-rmse:1541.75372
[254]   validation_0-rmse:1279.51300    validation_1-rmse:1541.28616
```

```
[255]   validation_0-rmse:1278.49747    validation_1-rmse:1541.30865
[256]   validation_0-rmse:1277.86545    validation_1-rmse:1541.19305
[257]   validation_0-rmse:1277.00470    validation_1-rmse:1541.00848
[258]   validation_0-rmse:1276.15887    validation_1-rmse:1540.53301
[259]   validation_0-rmse:1275.61041    validation_1-rmse:1540.52252
[260]   validation_0-rmse:1274.93867    validation_1-rmse:1540.18361
[261]   validation_0-rmse:1274.17132    validation_1-rmse:1539.89799
[262]   validation_0-rmse:1273.08165    validation_1-rmse:1539.65150
[263]   validation_0-rmse:1272.18047    validation_1-rmse:1539.24483
[264]   validation_0-rmse:1271.45907    validation_1-rmse:1538.68816
[265]   validation_0-rmse:1270.94639    validation_1-rmse:1538.87265
[266]   validation_0-rmse:1270.33166    validation_1-rmse:1538.73992
[267]   validation_0-rmse:1269.60379    validation_1-rmse:1538.68339
[268]   validation_0-rmse:1268.58696    validation_1-rmse:1538.19936
[269]   validation_0-rmse:1267.67057    validation_1-rmse:1537.74136
[270]   validation_0-rmse:1266.63954    validation_1-rmse:1537.66848
[271]   validation_0-rmse:1265.71961    validation_1-rmse:1536.97597
[272]   validation_0-rmse:1265.03086    validation_1-rmse:1536.67148
[273]   validation_0-rmse:1264.03207    validation_1-rmse:1536.11055
[274]   validation_0-rmse:1262.93190    validation_1-rmse:1535.74088
[275]   validation_0-rmse:1261.79965    validation_1-rmse:1535.13856
[276]   validation_0-rmse:1260.76210    validation_1-rmse:1534.64558
[277]   validation_0-rmse:1259.98913    validation_1-rmse:1534.33185
[278]   validation_0-rmse:1259.35848    validation_1-rmse:1533.87888
[279]   validation_0-rmse:1258.84586    validation_1-rmse:1533.40427
[280]   validation_0-rmse:1258.30234    validation_1-rmse:1533.19496
[281]   validation_0-rmse:1257.55275    validation_1-rmse:1532.93257
[282]   validation_0-rmse:1256.71248    validation_1-rmse:1532.69871
[283]   validation_0-rmse:1255.89980    validation_1-rmse:1532.28579
[284]   validation_0-rmse:1255.40671    validation_1-rmse:1532.01289
[285]   validation_0-rmse:1255.09025    validation_1-rmse:1532.13860
[286]   validation_0-rmse:1254.23984    validation_1-rmse:1531.63703
[287]   validation_0-rmse:1253.26855    validation_1-rmse:1531.76382
[288]   validation_0-rmse:1252.03009    validation_1-rmse:1531.03885
[289]   validation_0-rmse:1251.45917    validation_1-rmse:1531.41655
[290]   validation_0-rmse:1250.48471    validation_1-rmse:1530.90682
[291]   validation_0-rmse:1249.77745    validation_1-rmse:1530.58839
[292]   validation_0-rmse:1249.07283    validation_1-rmse:1530.51248
[293]   validation_0-rmse:1248.28949    validation_1-rmse:1529.83159
[294]   validation_0-rmse:1247.65579    validation_1-rmse:1529.55819
[295]   validation_0-rmse:1246.93417    validation_1-rmse:1529.21954
[296]   validation_0-rmse:1246.12656    validation_1-rmse:1529.08506
[297]   validation_0-rmse:1245.10158    validation_1-rmse:1528.93377
[298]   validation_0-rmse:1244.53877    validation_1-rmse:1528.80461
[299]   validation_0-rmse:1243.71235    validation_1-rmse:1528.85096
[300]   validation_0-rmse:1243.15153    validation_1-rmse:1528.82926
[301]   validation_0-rmse:1242.70761    validation_1-rmse:1528.72928
[302]   validation_0-rmse:1241.55179    validation_1-rmse:1528.04465
```

```
[303]   validation_0-rmse:1240.71381    validation_1-rmse:1527.69041
[304]   validation_0-rmse:1239.60432    validation_1-rmse:1527.29392
[305]   validation_0-rmse:1239.03504    validation_1-rmse:1527.36829
[306]   validation_0-rmse:1238.19435    validation_1-rmse:1527.25570
[307]   validation_0-rmse:1237.37962    validation_1-rmse:1526.84699
[308]   validation_0-rmse:1236.86699    validation_1-rmse:1526.87917
[309]   validation_0-rmse:1236.34731    validation_1-rmse:1526.84008
[310]   validation_0-rmse:1235.45921    validation_1-rmse:1526.56709
[311]   validation_0-rmse:1234.77452    validation_1-rmse:1526.25650
[312]   validation_0-rmse:1234.05338    validation_1-rmse:1526.21380
[313]   validation_0-rmse:1233.57666    validation_1-rmse:1525.96026
[314]   validation_0-rmse:1232.93444    validation_1-rmse:1525.52088
[315]   validation_0-rmse:1232.02708    validation_1-rmse:1525.30710
[316]   validation_0-rmse:1231.45311    validation_1-rmse:1524.92236
[317]   validation_0-rmse:1231.11767    validation_1-rmse:1524.85039
[318]   validation_0-rmse:1230.51107    validation_1-rmse:1524.56217
[319]   validation_0-rmse:1229.79935    validation_1-rmse:1524.41741
[320]   validation_0-rmse:1229.35800    validation_1-rmse:1524.28890
[321]   validation_0-rmse:1228.82895    validation_1-rmse:1524.13411
[322]   validation_0-rmse:1228.10438    validation_1-rmse:1524.04239
[323]   validation_0-rmse:1227.84682    validation_1-rmse:1524.19113
[324]   validation_0-rmse:1227.30770    validation_1-rmse:1524.24679
[325]   validation_0-rmse:1226.58923    validation_1-rmse:1524.29478
[326]   validation_0-rmse:1225.89645    validation_1-rmse:1523.90202
[327]   validation_0-rmse:1225.15279    validation_1-rmse:1523.64757
[328]   validation_0-rmse:1224.26873    validation_1-rmse:1523.54172
[329]   validation_0-rmse:1223.59882    validation_1-rmse:1523.55605
[330]   validation_0-rmse:1222.69018    validation_1-rmse:1523.49730
[331]   validation_0-rmse:1222.09324    validation_1-rmse:1523.63958
[332]   validation_0-rmse:1221.55404    validation_1-rmse:1523.61117
[333]   validation_0-rmse:1220.90785    validation_1-rmse:1523.40629
[334]   validation_0-rmse:1220.34092    validation_1-rmse:1523.17577
[335]   validation_0-rmse:1219.96594    validation_1-rmse:1522.99786
[336]   validation_0-rmse:1219.43914    validation_1-rmse:1522.87331
[337]   validation_0-rmse:1218.64584    validation_1-rmse:1522.77922
[338]   validation_0-rmse:1218.07203    validation_1-rmse:1522.48265
[339]   validation_0-rmse:1217.43447    validation_1-rmse:1522.30134
[340]   validation_0-rmse:1216.90941    validation_1-rmse:1522.12171
[341]   validation_0-rmse:1216.52060    validation_1-rmse:1522.19212
[342]   validation_0-rmse:1216.11151    validation_1-rmse:1521.99365
[343]   validation_0-rmse:1215.26399    validation_1-rmse:1521.40737
[344]   validation_0-rmse:1214.65087    validation_1-rmse:1520.93412
[345]   validation_0-rmse:1214.10862    validation_1-rmse:1520.75107
[346]   validation_0-rmse:1213.64734    validation_1-rmse:1520.50221
[347]   validation_0-rmse:1213.19434    validation_1-rmse:1520.58512
[348]   validation_0-rmse:1212.48120    validation_1-rmse:1520.45953
[349]   validation_0-rmse:1211.85357    validation_1-rmse:1520.41894
[350]   validation_0-rmse:1211.42713    validation_1-rmse:1520.43355
```

```
[351]   validation_0-rmse:1211.23657    validation_1-rmse:1520.19626
[352]   validation_0-rmse:1210.66895    validation_1-rmse:1519.95911
[353]   validation_0-rmse:1210.19976    validation_1-rmse:1520.01641
[354]   validation_0-rmse:1209.65070    validation_1-rmse:1519.95808
[355]   validation_0-rmse:1208.78076    validation_1-rmse:1519.56778
[356]   validation_0-rmse:1208.10394    validation_1-rmse:1519.26682
[357]   validation_0-rmse:1207.35303    validation_1-rmse:1519.05572
[358]   validation_0-rmse:1207.04344    validation_1-rmse:1519.19819
[359]   validation_0-rmse:1206.21748    validation_1-rmse:1518.94936
[360]   validation_0-rmse:1205.70157    validation_1-rmse:1518.74221
[361]   validation_0-rmse:1204.93986    validation_1-rmse:1518.90569
[362]   validation_0-rmse:1204.32162    validation_1-rmse:1518.96207
[363]   validation_0-rmse:1203.77626    validation_1-rmse:1518.83105
[364]   validation_0-rmse:1203.02091    validation_1-rmse:1518.75125
[365]   validation_0-rmse:1202.65790    validation_1-rmse:1518.62257
[366]   validation_0-rmse:1202.20273    validation_1-rmse:1518.53813
[367]   validation_0-rmse:1201.50696    validation_1-rmse:1518.48470
[368]   validation_0-rmse:1200.86183    validation_1-rmse:1518.51085
[369]   validation_0-rmse:1199.99638    validation_1-rmse:1518.25130
[370]   validation_0-rmse:1199.63136    validation_1-rmse:1518.11866
[371]   validation_0-rmse:1198.94861    validation_1-rmse:1517.85882
[372]   validation_0-rmse:1198.50029    validation_1-rmse:1517.70635
[373]   validation_0-rmse:1197.74786    validation_1-rmse:1517.33255
[374]   validation_0-rmse:1197.08443    validation_1-rmse:1517.29997
[375]   validation_0-rmse:1196.63310    validation_1-rmse:1517.19218
[376]   validation_0-rmse:1195.82202    validation_1-rmse:1516.76329
[377]   validation_0-rmse:1195.40679    validation_1-rmse:1516.97089
[378]   validation_0-rmse:1195.10595    validation_1-rmse:1516.88660
[379]   validation_0-rmse:1194.58642    validation_1-rmse:1517.08597
[380]   validation_0-rmse:1193.82282    validation_1-rmse:1516.95312
[381]   validation_0-rmse:1193.29935    validation_1-rmse:1516.49977
[382]   validation_0-rmse:1192.82766    validation_1-rmse:1516.16268
[383]   validation_0-rmse:1192.55832    validation_1-rmse:1516.28482
[384]   validation_0-rmse:1191.92518    validation_1-rmse:1516.15973
[385]   validation_0-rmse:1191.45163    validation_1-rmse:1516.00798
[386]   validation_0-rmse:1191.04811    validation_1-rmse:1516.23509
[387]   validation_0-rmse:1190.19799    validation_1-rmse:1515.96451
[388]   validation_0-rmse:1189.65664    validation_1-rmse:1515.89172
[389]   validation_0-rmse:1189.03149    validation_1-rmse:1515.73084
[390]   validation_0-rmse:1188.61662    validation_1-rmse:1515.56538
[391]   validation_0-rmse:1188.13170    validation_1-rmse:1515.26994
[392]   validation_0-rmse:1187.78472    validation_1-rmse:1515.51341
[393]   validation_0-rmse:1187.09109    validation_1-rmse:1515.27383
[394]   validation_0-rmse:1186.42138    validation_1-rmse:1515.00559
[395]   validation_0-rmse:1185.77223    validation_1-rmse:1514.79166
[396]   validation_0-rmse:1185.10605    validation_1-rmse:1514.45792
[397]   validation_0-rmse:1184.67395    validation_1-rmse:1514.50782
[398]   validation_0-rmse:1184.03546    validation_1-rmse:1514.66964
```

```
[399]   validation_0-rmse:1183.71586    validation_1-rmse:1514.77874
[400]   validation_0-rmse:1182.80826    validation_1-rmse:1514.43322
[401]   validation_0-rmse:1182.08939    validation_1-rmse:1514.27732
[402]   validation_0-rmse:1181.27722    validation_1-rmse:1514.02027
[403]   validation_0-rmse:1180.63553    validation_1-rmse:1513.63693
[404]   validation_0-rmse:1180.06708    validation_1-rmse:1513.54540
[405]   validation_0-rmse:1179.48427    validation_1-rmse:1513.40306
[406]   validation_0-rmse:1179.12389    validation_1-rmse:1513.42202
[407]   validation_0-rmse:1178.52875    validation_1-rmse:1513.43415
[408]   validation_0-rmse:1177.97372    validation_1-rmse:1513.35932
[409]   validation_0-rmse:1177.45627    validation_1-rmse:1513.04182
[410]   validation_0-rmse:1176.74248    validation_1-rmse:1512.92601
[411]   validation_0-rmse:1176.41979    validation_1-rmse:1513.03777
[412]   validation_0-rmse:1175.85241    validation_1-rmse:1512.69729
[413]   validation_0-rmse:1175.53065    validation_1-rmse:1512.65369
[414]   validation_0-rmse:1175.09165    validation_1-rmse:1512.64580
[415]   validation_0-rmse:1174.52241    validation_1-rmse:1512.21055
[416]   validation_0-rmse:1174.01249    validation_1-rmse:1512.30673
[417]   validation_0-rmse:1173.42211    validation_1-rmse:1512.20964
[418]   validation_0-rmse:1172.94780    validation_1-rmse:1512.21238
[419]   validation_0-rmse:1172.04657    validation_1-rmse:1511.69226
[420]   validation_0-rmse:1171.43212    validation_1-rmse:1511.44746
[421]   validation_0-rmse:1170.73228    validation_1-rmse:1511.11858
[422]   validation_0-rmse:1170.37488    validation_1-rmse:1510.92123
[423]   validation_0-rmse:1170.09421    validation_1-rmse:1510.82811
[424]   validation_0-rmse:1169.64886    validation_1-rmse:1510.87551
[425]   validation_0-rmse:1169.29826    validation_1-rmse:1510.96310
[426]   validation_0-rmse:1168.69311    validation_1-rmse:1510.78736
[427]   validation_0-rmse:1167.96215    validation_1-rmse:1510.84142
[428]   validation_0-rmse:1167.57844    validation_1-rmse:1510.94048
[429]   validation_0-rmse:1167.22648    validation_1-rmse:1511.13593
[430]   validation_0-rmse:1166.50961    validation_1-rmse:1510.98387
[431]   validation_0-rmse:1166.10504    validation_1-rmse:1510.81050
[432]   validation_0-rmse:1165.53231    validation_1-rmse:1510.62872
[433]   validation_0-rmse:1164.71172    validation_1-rmse:1510.20403
[434]   validation_0-rmse:1164.42955    validation_1-rmse:1510.30559
[435]   validation_0-rmse:1163.92704    validation_1-rmse:1509.94492
[436]   validation_0-rmse:1163.42352    validation_1-rmse:1509.98303
[437]   validation_0-rmse:1162.70320    validation_1-rmse:1509.61775
[438]   validation_0-rmse:1162.25239    validation_1-rmse:1509.78168
[439]   validation_0-rmse:1161.69950    validation_1-rmse:1509.79192
[440]   validation_0-rmse:1161.14426    validation_1-rmse:1509.50440
[441]   validation_0-rmse:1160.83453    validation_1-rmse:1509.56865
[442]   validation_0-rmse:1160.29396    validation_1-rmse:1509.27934
[443]   validation_0-rmse:1160.01209    validation_1-rmse:1509.22254
[444]   validation_0-rmse:1159.51006    validation_1-rmse:1509.18279
[445]   validation_0-rmse:1158.91903    validation_1-rmse:1509.14582
[446]   validation_0-rmse:1158.22113    validation_1-rmse:1508.77071
```

```
[447]   validation_0-rmse:1157.99162    validation_1-rmse:1508.61805
[448]   validation_0-rmse:1157.48511    validation_1-rmse:1508.39305
[449]   validation_0-rmse:1156.68803    validation_1-rmse:1507.85076
[450]   validation_0-rmse:1156.12435    validation_1-rmse:1507.57694
[451]   validation_0-rmse:1155.71084    validation_1-rmse:1507.64656
[452]   validation_0-rmse:1155.23167    validation_1-rmse:1507.56984
[453]   validation_0-rmse:1154.63620    validation_1-rmse:1507.32483
[454]   validation_0-rmse:1154.14380    validation_1-rmse:1507.33214
[455]   validation_0-rmse:1153.33894    validation_1-rmse:1506.95059
[456]   validation_0-rmse:1152.69400    validation_1-rmse:1506.89246
[457]   validation_0-rmse:1152.42324    validation_1-rmse:1506.89638
[458]   validation_0-rmse:1152.11580    validation_1-rmse:1507.00436
[459]   validation_0-rmse:1151.38841    validation_1-rmse:1506.77962
[460]   validation_0-rmse:1151.06207    validation_1-rmse:1506.69238
[461]   validation_0-rmse:1150.64454    validation_1-rmse:1506.58216
[462]   validation_0-rmse:1150.23235    validation_1-rmse:1506.52313
[463]   validation_0-rmse:1149.77227    validation_1-rmse:1506.19468
[464]   validation_0-rmse:1149.07529    validation_1-rmse:1506.01468
[465]   validation_0-rmse:1148.63470    validation_1-rmse:1505.89731
[466]   validation_0-rmse:1148.01900    validation_1-rmse:1505.71021
[467]   validation_0-rmse:1147.42126    validation_1-rmse:1505.60830
[468]   validation_0-rmse:1147.09747    validation_1-rmse:1505.60949
[469]   validation_0-rmse:1146.76489    validation_1-rmse:1505.68737
[470]   validation_0-rmse:1146.49141    validation_1-rmse:1505.51855
[471]   validation_0-rmse:1146.31603    validation_1-rmse:1505.54029
[472]   validation_0-rmse:1146.07367    validation_1-rmse:1505.50891
[473]   validation_0-rmse:1145.52394    validation_1-rmse:1505.49011
[474]   validation_0-rmse:1145.04450    validation_1-rmse:1505.41218
[475]   validation_0-rmse:1144.63341    validation_1-rmse:1505.59795
[476]   validation_0-rmse:1144.34712    validation_1-rmse:1505.49447
[477]   validation_0-rmse:1143.69946    validation_1-rmse:1505.30326
[478]   validation_0-rmse:1143.20285    validation_1-rmse:1505.41990
[479]   validation_0-rmse:1142.89987    validation_1-rmse:1505.37249
[480]   validation_0-rmse:1142.58009    validation_1-rmse:1505.35277
[481]   validation_0-rmse:1141.78093    validation_1-rmse:1505.11889
[482]   validation_0-rmse:1141.53455    validation_1-rmse:1505.01286
[483]   validation_0-rmse:1141.06145    validation_1-rmse:1504.75490
[484]   validation_0-rmse:1140.57826    validation_1-rmse:1504.49410
[485]   validation_0-rmse:1140.27958    validation_1-rmse:1504.57747
[486]   validation_0-rmse:1139.69775    validation_1-rmse:1504.27420
[487]   validation_0-rmse:1139.23192    validation_1-rmse:1504.33140
[488]   validation_0-rmse:1139.08023    validation_1-rmse:1504.38646
[489]   validation_0-rmse:1138.69704    validation_1-rmse:1504.41733
[490]   validation_0-rmse:1138.32225    validation_1-rmse:1504.34514
[491]   validation_0-rmse:1137.89063    validation_1-rmse:1504.45080
[492]   validation_0-rmse:1137.51774    validation_1-rmse:1504.55962
[493]   validation_0-rmse:1137.06087    validation_1-rmse:1504.45058
[494]   validation_0-rmse:1136.71878    validation_1-rmse:1504.53116
```

```
[495]   validation_0-rmse:1136.50092    validation_1-rmse:1504.47134
[496]   validation_0-rmse:1136.23778    validation_1-rmse:1504.46592
[497]   validation_0-rmse:1135.66690    validation_1-rmse:1504.56012
[498]   validation_0-rmse:1135.27494    validation_1-rmse:1504.59285
[499]   validation_0-rmse:1134.88704    validation_1-rmse:1504.93517
[500]   validation_0-rmse:1134.37763    validation_1-rmse:1504.71423
[501]   validation_0-rmse:1134.03287    validation_1-rmse:1504.51047
[502]   validation_0-rmse:1133.77039    validation_1-rmse:1504.59292
[503]   validation_0-rmse:1133.32122    validation_1-rmse:1504.68089
[504]   validation_0-rmse:1132.81249    validation_1-rmse:1504.51728
[505]   validation_0-rmse:1132.29969    validation_1-rmse:1504.57800
[506]   validation_0-rmse:1131.90719    validation_1-rmse:1504.44418
[507]   validation_0-rmse:1131.41154    validation_1-rmse:1504.55075
[508]   validation_0-rmse:1131.00705    validation_1-rmse:1504.53335
[509]   validation_0-rmse:1130.47037    validation_1-rmse:1504.41177
[510]   validation_0-rmse:1130.19810    validation_1-rmse:1504.29752
[511]   validation_0-rmse:1130.01141    validation_1-rmse:1504.17729
[512]   validation_0-rmse:1129.44262    validation_1-rmse:1503.99419
[513]   validation_0-rmse:1129.16317    validation_1-rmse:1504.11854
[514]   validation_0-rmse:1128.62393    validation_1-rmse:1503.80405
[515]   validation_0-rmse:1128.39327    validation_1-rmse:1503.93569
[516]   validation_0-rmse:1127.68251    validation_1-rmse:1503.64028
[517]   validation_0-rmse:1127.21060    validation_1-rmse:1503.63735
[518]   validation_0-rmse:1126.59171    validation_1-rmse:1503.31931
[519]   validation_0-rmse:1126.32577    validation_1-rmse:1503.25061
[520]   validation_0-rmse:1125.85613    validation_1-rmse:1503.18197
[521]   validation_0-rmse:1125.51058    validation_1-rmse:1503.14731
[522]   validation_0-rmse:1125.14659    validation_1-rmse:1503.21397
[523]   validation_0-rmse:1124.62345    validation_1-rmse:1503.02261
[524]   validation_0-rmse:1124.25810    validation_1-rmse:1502.98176
[525]   validation_0-rmse:1123.93669    validation_1-rmse:1503.20072
[526]   validation_0-rmse:1123.44136    validation_1-rmse:1503.14002
[527]   validation_0-rmse:1123.26071    validation_1-rmse:1503.46260
[528]   validation_0-rmse:1122.89463    validation_1-rmse:1503.31692
[529]   validation_0-rmse:1122.54771    validation_1-rmse:1503.22819
[530]   validation_0-rmse:1122.21468    validation_1-rmse:1503.10972
[531]   validation_0-rmse:1121.84183    validation_1-rmse:1503.13098
[532]   validation_0-rmse:1121.39737    validation_1-rmse:1503.13001
[533]   validation_0-rmse:1120.84679    validation_1-rmse:1502.87707
[534]   validation_0-rmse:1120.41217    validation_1-rmse:1502.75587
[535]   validation_0-rmse:1119.88905    validation_1-rmse:1502.47321
[536]   validation_0-rmse:1119.38194    validation_1-rmse:1502.40419
[537]   validation_0-rmse:1118.88314    validation_1-rmse:1502.28519
[538]   validation_0-rmse:1118.43266    validation_1-rmse:1502.11788
[539]   validation_0-rmse:1118.15134    validation_1-rmse:1502.03326
[540]   validation_0-rmse:1117.68600    validation_1-rmse:1501.89430
[541]   validation_0-rmse:1117.47809    validation_1-rmse:1502.06740
[542]   validation_0-rmse:1117.10014    validation_1-rmse:1502.09756
```

```
[543]   validation_0-rmse:1116.61818    validation_1-rmse:1502.12492
[544]   validation_0-rmse:1116.36731    validation_1-rmse:1502.14746
[545]   validation_0-rmse:1115.88171    validation_1-rmse:1501.85914
[546]   validation_0-rmse:1115.27541    validation_1-rmse:1501.74316
[547]   validation_0-rmse:1114.84635    validation_1-rmse:1501.88632
[548]   validation_0-rmse:1114.60622    validation_1-rmse:1501.91199
[549]   validation_0-rmse:1114.24474    validation_1-rmse:1502.05704
[550]   validation_0-rmse:1113.97245    validation_1-rmse:1502.38298
[551]   validation_0-rmse:1113.75488    validation_1-rmse:1502.50822
[552]   validation_0-rmse:1113.55701    validation_1-rmse:1502.22914
[553]   validation_0-rmse:1113.15877    validation_1-rmse:1502.27931
[554]   validation_0-rmse:1112.61638    validation_1-rmse:1502.01877
[555]   validation_0-rmse:1112.11740    validation_1-rmse:1502.14316
[556]   validation_0-rmse:1111.79326    validation_1-rmse:1502.18550
[557]   validation_0-rmse:1111.35993    validation_1-rmse:1502.06698
[558]   validation_0-rmse:1110.98552    validation_1-rmse:1502.02471
[559]   validation_0-rmse:1110.65274    validation_1-rmse:1501.94562
[560]   validation_0-rmse:1110.28129    validation_1-rmse:1501.82869
[561]   validation_0-rmse:1109.76397    validation_1-rmse:1501.77212
[562]   validation_0-rmse:1109.35495    validation_1-rmse:1501.95665
[563]   validation_0-rmse:1109.13376    validation_1-rmse:1502.04715
[564]   validation_0-rmse:1108.73472    validation_1-rmse:1501.96468
[565]   validation_0-rmse:1108.33124    validation_1-rmse:1501.78793
[566]   validation_0-rmse:1107.98426    validation_1-rmse:1501.88118
[567]   validation_0-rmse:1107.83760    validation_1-rmse:1501.82648
[568]   validation_0-rmse:1107.57574    validation_1-rmse:1501.82022
[569]   validation_0-rmse:1107.05042    validation_1-rmse:1501.74238
[570]   validation_0-rmse:1106.83575    validation_1-rmse:1501.79518
[571]   validation_0-rmse:1106.48156    validation_1-rmse:1502.01236
[572]   validation_0-rmse:1105.99304    validation_1-rmse:1502.18235
[573]   validation_0-rmse:1105.55757    validation_1-rmse:1502.25562
[574]   validation_0-rmse:1105.12387    validation_1-rmse:1502.43398
[575]   validation_0-rmse:1104.82286    validation_1-rmse:1502.53651
[576]   validation_0-rmse:1104.58803    validation_1-rmse:1502.71801
[577]   validation_0-rmse:1104.34410    validation_1-rmse:1502.66083
[578]   validation_0-rmse:1104.13948    validation_1-rmse:1502.74741
[579]   validation_0-rmse:1103.72814    validation_1-rmse:1502.81216
[580]   validation_0-rmse:1103.57167    validation_1-rmse:1502.80125
[581]   validation_0-rmse:1103.20060    validation_1-rmse:1502.95934
[582]   validation_0-rmse:1102.59573    validation_1-rmse:1502.60765
[583]   validation_0-rmse:1102.20857    validation_1-rmse:1502.62211
[584]   validation_0-rmse:1101.95867    validation_1-rmse:1502.70632
[585]   validation_0-rmse:1101.61316    validation_1-rmse:1502.86282
[586]   validation_0-rmse:1101.19942    validation_1-rmse:1502.65640
[587]   validation_0-rmse:1100.77140    validation_1-rmse:1502.64737
[588]   validation_0-rmse:1100.36144    validation_1-rmse:1502.58118
[589]   validation_0-rmse:1099.56898    validation_1-rmse:1502.23988
[590]   validation_0-rmse:1099.14191    validation_1-rmse:1502.25793
```

```
[591]   validation_0-rmse:1098.73299    validation_1-rmse:1502.22320
[592]   validation_0-rmse:1098.25421    validation_1-rmse:1502.25324
[593]   validation_0-rmse:1097.83417    validation_1-rmse:1502.01438
[594]   validation_0-rmse:1097.33848    validation_1-rmse:1501.93306
[595]   validation_0-rmse:1096.99406    validation_1-rmse:1501.85513
[596]   validation_0-rmse:1096.64927    validation_1-rmse:1501.97459
[597]   validation_0-rmse:1096.43345    validation_1-rmse:1501.91004
[598]   validation_0-rmse:1096.03504    validation_1-rmse:1501.92410
[599]   validation_0-rmse:1095.80773    validation_1-rmse:1501.91005
[600]   validation_0-rmse:1095.39663    validation_1-rmse:1501.88145
[601]   validation_0-rmse:1094.96699    validation_1-rmse:1501.87816
[602]   validation_0-rmse:1094.71097    validation_1-rmse:1501.79114
[603]   validation_0-rmse:1094.32354    validation_1-rmse:1501.87875
[604]   validation_0-rmse:1094.17629    validation_1-rmse:1501.83925
[605]   validation_0-rmse:1093.91589    validation_1-rmse:1502.08686
[606]   validation_0-rmse:1093.59437    validation_1-rmse:1502.08723
[607]   validation_0-rmse:1093.27189    validation_1-rmse:1502.20743
[608]   validation_0-rmse:1092.91976    validation_1-rmse:1502.05035
[609]   validation_0-rmse:1092.58517    validation_1-rmse:1502.06832
[610]   validation_0-rmse:1092.22122    validation_1-rmse:1501.90012
[611]   validation_0-rmse:1091.94879    validation_1-rmse:1501.87016
[612]   validation_0-rmse:1091.57356    validation_1-rmse:1501.67927
[613]   validation_0-rmse:1091.34541    validation_1-rmse:1501.69930
[614]   validation_0-rmse:1091.03415    validation_1-rmse:1501.76714
[615]   validation_0-rmse:1090.68416    validation_1-rmse:1501.84050
[616]   validation_0-rmse:1090.35736    validation_1-rmse:1501.82790
[617]   validation_0-rmse:1090.18621    validation_1-rmse:1501.93337
[618]   validation_0-rmse:1089.82469    validation_1-rmse:1502.13203
[619]   validation_0-rmse:1089.41934    validation_1-rmse:1502.26482
[620]   validation_0-rmse:1089.25066    validation_1-rmse:1502.21049
[621]   validation_0-rmse:1088.98902    validation_1-rmse:1502.28339
[622]   validation_0-rmse:1088.61145    validation_1-rmse:1502.30741
[623]   validation_0-rmse:1088.11523    validation_1-rmse:1502.29169
[624]   validation_0-rmse:1087.83510    validation_1-rmse:1502.33600
[625]   validation_0-rmse:1087.64713    validation_1-rmse:1502.37018
[626]   validation_0-rmse:1087.47284    validation_1-rmse:1502.35504
[627]   validation_0-rmse:1087.18355    validation_1-rmse:1502.43191
[628]   validation_0-rmse:1086.99842    validation_1-rmse:1502.34180
[629]   validation_0-rmse:1086.77473    validation_1-rmse:1502.27651
[630]   validation_0-rmse:1086.43201    validation_1-rmse:1502.23874
[631]   validation_0-rmse:1086.30586    validation_1-rmse:1502.37298
[632]   validation_0-rmse:1086.21699    validation_1-rmse:1502.35877
[633]   validation_0-rmse:1085.99213    validation_1-rmse:1502.29752
[634]   validation_0-rmse:1085.72453    validation_1-rmse:1502.44342
[635]   validation_0-rmse:1085.63585    validation_1-rmse:1502.48037
[636]   validation_0-rmse:1085.48195    validation_1-rmse:1502.40929
[637]   validation_0-rmse:1085.27717    validation_1-rmse:1502.50221
[638]   validation_0-rmse:1084.95596    validation_1-rmse:1502.39633
```

```
[639]   validation_0-rmse:1084.62496    validation_1-rmse:1502.39289
[640]   validation_0-rmse:1084.22149    validation_1-rmse:1502.32760
[641]   validation_0-rmse:1083.69205    validation_1-rmse:1502.18335
[642]   validation_0-rmse:1083.41295    validation_1-rmse:1502.00854
[643]   validation_0-rmse:1083.13002    validation_1-rmse:1501.89357
[644]   validation_0-rmse:1082.89845    validation_1-rmse:1502.25562
[645]   validation_0-rmse:1082.57255    validation_1-rmse:1502.35126
[646]   validation_0-rmse:1082.09577    validation_1-rmse:1502.24863
[647]   validation_0-rmse:1081.75525    validation_1-rmse:1502.00344
[648]   validation_0-rmse:1081.41438    validation_1-rmse:1501.94243
[649]   validation_0-rmse:1080.89351    validation_1-rmse:1501.82985
[650]   validation_0-rmse:1080.40821    validation_1-rmse:1501.69219
[651]   validation_0-rmse:1080.20217    validation_1-rmse:1501.85212
[652]   validation_0-rmse:1080.08202    validation_1-rmse:1501.91081
[653]   validation_0-rmse:1079.72016    validation_1-rmse:1501.87953
[654]   validation_0-rmse:1079.55188    validation_1-rmse:1501.71008
[655]   validation_0-rmse:1079.07806    validation_1-rmse:1501.68788
[656]   validation_0-rmse:1078.70546    validation_1-rmse:1501.65795
[657]   validation_0-rmse:1078.60220    validation_1-rmse:1501.56551
[658]   validation_0-rmse:1078.35450    validation_1-rmse:1501.52654
[659]   validation_0-rmse:1078.26368    validation_1-rmse:1501.55357
[660]   validation_0-rmse:1078.06324    validation_1-rmse:1501.43986
[661]   validation_0-rmse:1077.87309    validation_1-rmse:1501.57562
[662]   validation_0-rmse:1077.56766    validation_1-rmse:1501.55905
[663]   validation_0-rmse:1077.36306    validation_1-rmse:1501.57477
[664]   validation_0-rmse:1077.03807    validation_1-rmse:1501.58253
[665]   validation_0-rmse:1076.82121    validation_1-rmse:1501.64444
[666]   validation_0-rmse:1076.51930    validation_1-rmse:1501.69477
[667]   validation_0-rmse:1076.10125    validation_1-rmse:1501.55591
[668]   validation_0-rmse:1075.76425    validation_1-rmse:1501.56316
[669]   validation_0-rmse:1075.48440    validation_1-rmse:1501.63925
[670]   validation_0-rmse:1075.19810    validation_1-rmse:1501.64376
[671]   validation_0-rmse:1074.86528    validation_1-rmse:1501.71364
[672]   validation_0-rmse:1074.60564    validation_1-rmse:1501.69250
[673]   validation_0-rmse:1074.23652    validation_1-rmse:1501.58629
[674]   validation_0-rmse:1074.03707    validation_1-rmse:1501.65871
[675]   validation_0-rmse:1073.67649    validation_1-rmse:1501.48951
[676]   validation_0-rmse:1073.37025    validation_1-rmse:1501.62048
[677]   validation_0-rmse:1073.20646    validation_1-rmse:1501.48979
[678]   validation_0-rmse:1073.04059    validation_1-rmse:1501.34694
[679]   validation_0-rmse:1072.89121    validation_1-rmse:1501.31374
[680]   validation_0-rmse:1072.64930    validation_1-rmse:1501.51056
[681]   validation_0-rmse:1072.45948    validation_1-rmse:1501.82170
[682]   validation_0-rmse:1071.99831    validation_1-rmse:1501.74450
[683]   validation_0-rmse:1071.83332    validation_1-rmse:1501.72730
[684]   validation_0-rmse:1071.54805    validation_1-rmse:1501.79709
[685]   validation_0-rmse:1071.13504    validation_1-rmse:1501.63615
[686]   validation_0-rmse:1070.78356    validation_1-rmse:1501.45967
```

```
[687]   validation_0-rmse:1070.63412    validation_1-rmse:1501.75729
[688]   validation_0-rmse:1070.46765    validation_1-rmse:1501.89180
[689]   validation_0-rmse:1069.99867    validation_1-rmse:1501.88147
[690]   validation_0-rmse:1069.93500    validation_1-rmse:1501.87209
[691]   validation_0-rmse:1069.72564    validation_1-rmse:1501.89759
[692]   validation_0-rmse:1069.46716    validation_1-rmse:1501.96446
[693]   validation_0-rmse:1069.29436    validation_1-rmse:1501.96227
[694]   validation_0-rmse:1069.06042    validation_1-rmse:1501.99662
[695]   validation_0-rmse:1068.88804    validation_1-rmse:1502.10289
[696]   validation_0-rmse:1068.58209    validation_1-rmse:1502.06300
[697]   validation_0-rmse:1068.12967    validation_1-rmse:1502.03279
[698]   validation_0-rmse:1067.98020    validation_1-rmse:1501.99439
[699]   validation_0-rmse:1067.79035    validation_1-rmse:1502.15199
[700]   validation_0-rmse:1067.64991    validation_1-rmse:1501.86929
[701]   validation_0-rmse:1067.15344    validation_1-rmse:1501.88136
[702]   validation_0-rmse:1066.98563    validation_1-rmse:1501.91785
[703]   validation_0-rmse:1066.67187    validation_1-rmse:1502.05614
[704]   validation_0-rmse:1066.43688    validation_1-rmse:1501.99411
[705]   validation_0-rmse:1066.18556    validation_1-rmse:1502.11120
[706]   validation_0-rmse:1066.01653    validation_1-rmse:1502.16050
[707]   validation_0-rmse:1065.70331    validation_1-rmse:1502.15357
[708]   validation_0-rmse:1065.32810    validation_1-rmse:1502.03566
[709]   validation_0-rmse:1064.89189    validation_1-rmse:1502.08299
[710]   validation_0-rmse:1064.61086    validation_1-rmse:1502.26084
[711]   validation_0-rmse:1064.09121    validation_1-rmse:1502.20243
[712]   validation_0-rmse:1063.86731    validation_1-rmse:1502.20920
[713]   validation_0-rmse:1063.40933    validation_1-rmse:1502.18888
[714]   validation_0-rmse:1063.28870    validation_1-rmse:1502.22873
[715]   validation_0-rmse:1063.11308    validation_1-rmse:1502.35443
[716]   validation_0-rmse:1062.87700    validation_1-rmse:1502.39328
[717]   validation_0-rmse:1062.66681    validation_1-rmse:1502.28764
[718]   validation_0-rmse:1062.48258    validation_1-rmse:1502.33443
[719]   validation_0-rmse:1062.19638    validation_1-rmse:1502.35403
[720]   validation_0-rmse:1061.89287    validation_1-rmse:1502.51317
[721]   validation_0-rmse:1061.66589    validation_1-rmse:1502.59649
[722]   validation_0-rmse:1061.43569    validation_1-rmse:1502.52693
[723]   validation_0-rmse:1061.26976    validation_1-rmse:1502.57633
[724]   validation_0-rmse:1060.95912    validation_1-rmse:1502.44588
[725]   validation_0-rmse:1060.64070    validation_1-rmse:1502.44389
[726]   validation_0-rmse:1060.31842    validation_1-rmse:1502.49752
[727]   validation_0-rmse:1060.15785    validation_1-rmse:1502.49197
[728]   validation_0-rmse:1060.00339    validation_1-rmse:1502.66428
[729]   validation_0-rmse:1059.70266    validation_1-rmse:1502.57631
[730]   validation_0-rmse:1059.52595    validation_1-rmse:1502.61112
[731]   validation_0-rmse:1059.31635    validation_1-rmse:1502.66922
[732]   validation_0-rmse:1059.01815    validation_1-rmse:1502.69234
[733]   validation_0-rmse:1058.91986    validation_1-rmse:1502.71633
[734]   validation_0-rmse:1058.66672    validation_1-rmse:1502.64535
```

```
[735]   validation_0-rmse:1058.38932    validation_1-rmse:1502.58217
[736]   validation_0-rmse:1058.14010    validation_1-rmse:1502.68729
[737]   validation_0-rmse:1058.06514    validation_1-rmse:1502.78583
[738]   validation_0-rmse:1057.75751    validation_1-rmse:1502.67715
[739]   validation_0-rmse:1057.55473    validation_1-rmse:1502.63618
[740]   validation_0-rmse:1057.03348    validation_1-rmse:1502.50009
[741]   validation_0-rmse:1056.73232    validation_1-rmse:1502.54361
[742]   validation_0-rmse:1056.34555    validation_1-rmse:1502.40831
[743]   validation_0-rmse:1056.16302    validation_1-rmse:1502.37879
[744]   validation_0-rmse:1055.92459    validation_1-rmse:1502.43269
[745]   validation_0-rmse:1055.88215    validation_1-rmse:1502.29590
[746]   validation_0-rmse:1055.51940    validation_1-rmse:1502.26820
[747]   validation_0-rmse:1054.90755    validation_1-rmse:1502.06655
[748]   validation_0-rmse:1054.59452    validation_1-rmse:1502.09593
[749]   validation_0-rmse:1054.40579    validation_1-rmse:1501.98361
[750]   validation_0-rmse:1054.07130    validation_1-rmse:1501.99973
[751]   validation_0-rmse:1053.86909    validation_1-rmse:1502.04379
[752]   validation_0-rmse:1053.82615    validation_1-rmse:1501.80881
[753]   validation_0-rmse:1053.60334    validation_1-rmse:1501.69692
[754]   validation_0-rmse:1053.42776    validation_1-rmse:1501.53208
[755]   validation_0-rmse:1053.21459    validation_1-rmse:1501.52813
[756]   validation_0-rmse:1052.92334    validation_1-rmse:1501.52517
[757]   validation_0-rmse:1052.67543    validation_1-rmse:1501.46982
[758]   validation_0-rmse:1052.45302    validation_1-rmse:1501.54290
[759]   validation_0-rmse:1052.09214    validation_1-rmse:1501.47412
[760]   validation_0-rmse:1051.92509    validation_1-rmse:1501.57783
[761]   validation_0-rmse:1051.69223    validation_1-rmse:1501.51433
[762]   validation_0-rmse:1051.48366    validation_1-rmse:1501.46684
[763]   validation_0-rmse:1051.39770    validation_1-rmse:1501.49227
[764]   validation_0-rmse:1051.19534    validation_1-rmse:1501.33118
[765]   validation_0-rmse:1050.95837    validation_1-rmse:1501.17595
[766]   validation_0-rmse:1050.85839    validation_1-rmse:1501.29335
[767]   validation_0-rmse:1050.76843    validation_1-rmse:1501.39700
[768]   validation_0-rmse:1050.57244    validation_1-rmse:1501.39702
[769]   validation_0-rmse:1050.28505    validation_1-rmse:1501.40180
[770]   validation_0-rmse:1050.07731    validation_1-rmse:1501.56274
[771]   validation_0-rmse:1049.85057    validation_1-rmse:1501.65268
[772]   validation_0-rmse:1049.71182    validation_1-rmse:1501.73500
[773]   validation_0-rmse:1049.39847    validation_1-rmse:1501.71758
[774]   validation_0-rmse:1048.99406    validation_1-rmse:1501.79819
[775]   validation_0-rmse:1048.77279    validation_1-rmse:1501.88784
[776]   validation_0-rmse:1048.60813    validation_1-rmse:1501.90029
[777]   validation_0-rmse:1048.33018    validation_1-rmse:1501.90430
[778]   validation_0-rmse:1048.02351    validation_1-rmse:1501.84600
[779]   validation_0-rmse:1047.92889    validation_1-rmse:1501.70633
[780]   validation_0-rmse:1047.82716    validation_1-rmse:1501.74524
[781]   validation_0-rmse:1047.72093    validation_1-rmse:1501.67783
[782]   validation_0-rmse:1047.51256    validation_1-rmse:1501.90053
```

```
[783]   validation_0-rmse:1047.43421    validation_1-rmse:1501.97443
[784]   validation_0-rmse:1047.20245    validation_1-rmse:1502.08408
[785]   validation_0-rmse:1046.70605    validation_1-rmse:1502.17098
[786]   validation_0-rmse:1046.54207    validation_1-rmse:1502.27249
[787]   validation_0-rmse:1046.24607    validation_1-rmse:1502.23289
[788]   validation_0-rmse:1046.05428    validation_1-rmse:1502.31840
[789]   validation_0-rmse:1045.79365    validation_1-rmse:1502.19832
[790]   validation_0-rmse:1045.66221    validation_1-rmse:1502.21082
[791]   validation_0-rmse:1045.26573    validation_1-rmse:1502.08373
[792]   validation_0-rmse:1045.11327    validation_1-rmse:1502.14102
[793]   validation_0-rmse:1044.92962    validation_1-rmse:1502.27972
[794]   validation_0-rmse:1044.64580    validation_1-rmse:1502.27377
[795]   validation_0-rmse:1044.22299    validation_1-rmse:1502.18710
[796]   validation_0-rmse:1043.85428    validation_1-rmse:1502.08086
[797]   validation_0-rmse:1043.59002    validation_1-rmse:1502.24545
[798]   validation_0-rmse:1043.32290    validation_1-rmse:1502.09072
[799]   validation_0-rmse:1043.06491    validation_1-rmse:1502.07019
[800]   validation_0-rmse:1042.82701    validation_1-rmse:1502.10340
[801]   validation_0-rmse:1042.69655    validation_1-rmse:1502.11772
[802]   validation_0-rmse:1042.40062    validation_1-rmse:1502.23860
[803]   validation_0-rmse:1042.15621    validation_1-rmse:1502.20331
[804]   validation_0-rmse:1041.96855    validation_1-rmse:1502.39051
[805]   validation_0-rmse:1041.77439    validation_1-rmse:1502.42513
[806]   validation_0-rmse:1041.53637    validation_1-rmse:1502.42010
[807]   validation_0-rmse:1041.27506    validation_1-rmse:1502.58623
[808]   validation_0-rmse:1041.00703    validation_1-rmse:1502.49896
[809]   validation_0-rmse:1040.80709    validation_1-rmse:1502.35920
[810]   validation_0-rmse:1040.60971    validation_1-rmse:1502.34323
[811]   validation_0-rmse:1040.41296    validation_1-rmse:1502.41553
[812]   validation_0-rmse:1039.99993    validation_1-rmse:1502.25861
[813]   validation_0-rmse:1039.70974    validation_1-rmse:1502.26490
[814]   validation_0-rmse:1039.49762    validation_1-rmse:1502.30929
[815]   validation_0-rmse:1039.29250    validation_1-rmse:1502.20220
[816]   validation_0-rmse:1039.08955    validation_1-rmse:1502.22574
[817]   validation_0-rmse:1038.85390    validation_1-rmse:1502.20485
[818]   validation_0-rmse:1038.74862    validation_1-rmse:1502.17779
[819]   validation_0-rmse:1038.61737    validation_1-rmse:1502.22089
[820]   validation_0-rmse:1038.49880    validation_1-rmse:1502.51953
[821]   validation_0-rmse:1038.28892    validation_1-rmse:1502.55284
[822]   validation_0-rmse:1038.12554    validation_1-rmse:1502.67854
[823]   validation_0-rmse:1037.82755    validation_1-rmse:1502.64949
[824]   validation_0-rmse:1037.52430    validation_1-rmse:1502.72790
[825]   validation_0-rmse:1037.28858    validation_1-rmse:1502.95148
[826]   validation_0-rmse:1037.10258    validation_1-rmse:1502.91200
[827]   validation_0-rmse:1036.86954    validation_1-rmse:1503.01021
[828]   validation_0-rmse:1036.76146    validation_1-rmse:1503.02743
[829]   validation_0-rmse:1036.50434    validation_1-rmse:1503.07592
[830]   validation_0-rmse:1036.36417    validation_1-rmse:1503.20106
```

```
[831]   validation_0-rmse:1036.25742    validation_1-rmse:1503.24841
[832]   validation_0-rmse:1035.99173    validation_1-rmse:1503.18935
[833]   validation_0-rmse:1035.85603    validation_1-rmse:1503.36955
[834]   validation_0-rmse:1035.63383    validation_1-rmse:1503.32356
[835]   validation_0-rmse:1035.50350    validation_1-rmse:1503.37229
[836]   validation_0-rmse:1035.38085    validation_1-rmse:1503.30570
[837]   validation_0-rmse:1035.13799    validation_1-rmse:1503.41426
[838]   validation_0-rmse:1034.90705    validation_1-rmse:1503.54919
[839]   validation_0-rmse:1034.81962    validation_1-rmse:1503.49312
[840]   validation_0-rmse:1034.60509    validation_1-rmse:1503.51887
[841]   validation_0-rmse:1034.38952    validation_1-rmse:1503.52353
[842]   validation_0-rmse:1034.31005    validation_1-rmse:1503.55922
[843]   validation_0-rmse:1034.01918    validation_1-rmse:1503.60439
[844]   validation_0-rmse:1033.74566    validation_1-rmse:1503.68690
[845]   validation_0-rmse:1033.57291    validation_1-rmse:1503.84248
[846]   validation_0-rmse:1033.50892    validation_1-rmse:1503.87490
[847]   validation_0-rmse:1033.24621    validation_1-rmse:1503.84543
[848]   validation_0-rmse:1033.00132    validation_1-rmse:1503.91148
[849]   validation_0-rmse:1032.90021    validation_1-rmse:1503.95390
[850]   validation_0-rmse:1032.75938    validation_1-rmse:1503.95546
[851]   validation_0-rmse:1032.60999    validation_1-rmse:1503.89261
[852]   validation_0-rmse:1032.53096    validation_1-rmse:1503.87824
[853]   validation_0-rmse:1032.43534    validation_1-rmse:1503.92359
[854]   validation_0-rmse:1032.22138    validation_1-rmse:1503.78074
[855]   validation_0-rmse:1032.06284    validation_1-rmse:1503.82836
[856]   validation_0-rmse:1031.68821    validation_1-rmse:1503.65255
[857]   validation_0-rmse:1031.54854    validation_1-rmse:1503.65140
[858]   validation_0-rmse:1031.23089    validation_1-rmse:1503.77307
[859]   validation_0-rmse:1031.09779    validation_1-rmse:1503.83556
[860]   validation_0-rmse:1031.02690    validation_1-rmse:1503.80429
[861]   validation_0-rmse:1030.73786    validation_1-rmse:1503.91753
[862]   validation_0-rmse:1030.43793    validation_1-rmse:1503.90319
[863]   validation_0-rmse:1030.31777    validation_1-rmse:1503.84522
[864]   validation_0-rmse:1030.01402    validation_1-rmse:1503.85768
[865]   validation_0-rmse:1029.89929    validation_1-rmse:1503.93304
[866]   validation_0-rmse:1029.64345    validation_1-rmse:1503.79402
[867]   validation_0-rmse:1029.35281    validation_1-rmse:1503.95791
[868]   validation_0-rmse:1029.14432    validation_1-rmse:1503.86884
[869]   validation_0-rmse:1028.92629    validation_1-rmse:1503.85707
[870]   validation_0-rmse:1028.83374    validation_1-rmse:1503.84382
[871]   validation_0-rmse:1028.72503    validation_1-rmse:1503.76383
[872]   validation_0-rmse:1028.51384    validation_1-rmse:1503.69401
[873]   validation_0-rmse:1028.22397    validation_1-rmse:1503.79794
[874]   validation_0-rmse:1027.90377    validation_1-rmse:1503.78245
[875]   validation_0-rmse:1027.63614    validation_1-rmse:1503.77275
[876]   validation_0-rmse:1027.49700    validation_1-rmse:1503.72181
[877]   validation_0-rmse:1027.39206    validation_1-rmse:1503.44281
[878]   validation_0-rmse:1027.16225    validation_1-rmse:1503.51594
```

```
[879]   validation_0-rmse:1026.95776    validation_1-rmse:1503.63009
[880]   validation_0-rmse:1026.86513    validation_1-rmse:1503.62060
[881]   validation_0-rmse:1026.76806    validation_1-rmse:1503.64841
[882]   validation_0-rmse:1026.67702    validation_1-rmse:1503.56284
[883]   validation_0-rmse:1026.39839    validation_1-rmse:1503.46071
[884]   validation_0-rmse:1026.04587    validation_1-rmse:1503.35957
[885]   validation_0-rmse:1025.88494    validation_1-rmse:1503.30776
[886]   validation_0-rmse:1025.58769    validation_1-rmse:1503.24619
[887]   validation_0-rmse:1025.41965    validation_1-rmse:1503.33570
[888]   validation_0-rmse:1025.22932    validation_1-rmse:1503.46258
[889]   validation_0-rmse:1025.04589    validation_1-rmse:1502.93816
[890]   validation_0-rmse:1024.83908    validation_1-rmse:1502.79963
[891]   validation_0-rmse:1024.61179    validation_1-rmse:1502.93410
[892]   validation_0-rmse:1024.37651    validation_1-rmse:1502.83410
[893]   validation_0-rmse:1024.23628    validation_1-rmse:1503.04008
[894]   validation_0-rmse:1023.81525    validation_1-rmse:1503.10481
[895]   validation_0-rmse:1023.63591    validation_1-rmse:1503.03939
[896]   validation_0-rmse:1023.49325    validation_1-rmse:1502.98149
[897]   validation_0-rmse:1023.38010    validation_1-rmse:1503.04481
[898]   validation_0-rmse:1023.17584    validation_1-rmse:1502.89715
[899]   validation_0-rmse:1022.91651    validation_1-rmse:1502.96384
[900]   validation_0-rmse:1022.58378    validation_1-rmse:1502.85540
[901]   validation_0-rmse:1022.46437    validation_1-rmse:1502.89058
[902]   validation_0-rmse:1022.23846    validation_1-rmse:1503.06882
[903]   validation_0-rmse:1022.14423    validation_1-rmse:1503.09808
[904]   validation_0-rmse:1021.86914    validation_1-rmse:1503.08141
[905]   validation_0-rmse:1021.61947    validation_1-rmse:1503.01529
[906]   validation_0-rmse:1021.42496    validation_1-rmse:1502.94305
[907]   validation_0-rmse:1021.19008    validation_1-rmse:1503.00753
[908]   validation_0-rmse:1020.95263    validation_1-rmse:1503.16260
[909]   validation_0-rmse:1020.80249    validation_1-rmse:1503.33208
[910]   validation_0-rmse:1020.68317    validation_1-rmse:1503.28151
[911]   validation_0-rmse:1020.47250    validation_1-rmse:1503.24744
[912]   validation_0-rmse:1020.32684    validation_1-rmse:1503.17169
[913]   validation_0-rmse:1020.13678    validation_1-rmse:1503.37544
[914]   validation_0-rmse:1019.99088    validation_1-rmse:1503.39085
[915]   validation_0-rmse:1019.80647    validation_1-rmse:1503.48824
[916]   validation_0-rmse:1019.58327    validation_1-rmse:1503.48622
[917]   validation_0-rmse:1019.49153    validation_1-rmse:1503.51217
[918]   validation_0-rmse:1019.40192    validation_1-rmse:1503.55566
[919]   validation_0-rmse:1019.09099    validation_1-rmse:1503.69040
[920]   validation_0-rmse:1018.75947    validation_1-rmse:1503.43447
[921]   validation_0-rmse:1018.56011    validation_1-rmse:1503.60132
[922]   validation_0-rmse:1018.44796    validation_1-rmse:1503.61249
[923]   validation_0-rmse:1018.15377    validation_1-rmse:1503.52896
[924]   validation_0-rmse:1017.98490    validation_1-rmse:1503.56991
[925]   validation_0-rmse:1017.84644    validation_1-rmse:1503.44842
[926]   validation_0-rmse:1017.69083    validation_1-rmse:1503.43992
```

```
[927]   validation_0-rmse:1017.62196    validation_1-rmse:1503.59509
[928]   validation_0-rmse:1017.59525    validation_1-rmse:1503.57415
[929]   validation_0-rmse:1017.49285    validation_1-rmse:1503.66578
[930]   validation_0-rmse:1017.32059    validation_1-rmse:1503.76273
[931]   validation_0-rmse:1017.18479    validation_1-rmse:1503.85555
[932]   validation_0-rmse:1016.90738    validation_1-rmse:1503.85009
[933]   validation_0-rmse:1016.66389    validation_1-rmse:1504.03163
[934]   validation_0-rmse:1016.46165    validation_1-rmse:1503.96641
[935]   validation_0-rmse:1016.24608    validation_1-rmse:1504.11372
[936]   validation_0-rmse:1016.07892    validation_1-rmse:1504.12008
[937]   validation_0-rmse:1015.96217    validation_1-rmse:1504.12989
[938]   validation_0-rmse:1015.76036    validation_1-rmse:1504.15798
[939]   validation_0-rmse:1015.49960    validation_1-rmse:1504.08882
[940]   validation_0-rmse:1015.34928    validation_1-rmse:1504.16815
[941]   validation_0-rmse:1015.25468    validation_1-rmse:1504.18211
[942]   validation_0-rmse:1015.04752    validation_1-rmse:1504.14292
[943]   validation_0-rmse:1014.75416    validation_1-rmse:1504.10468
[944]   validation_0-rmse:1014.43378    validation_1-rmse:1504.12666
[945]   validation_0-rmse:1014.11925    validation_1-rmse:1504.25281
[946]   validation_0-rmse:1013.98188    validation_1-rmse:1504.31294
[947]   validation_0-rmse:1013.73525    validation_1-rmse:1504.21200
[948]   validation_0-rmse:1013.46831    validation_1-rmse:1504.17972
[949]   validation_0-rmse:1013.34674    validation_1-rmse:1504.29683
[950]   validation_0-rmse:1013.20479    validation_1-rmse:1504.45880
[951]   validation_0-rmse:1012.97047    validation_1-rmse:1504.42109
[952]   validation_0-rmse:1012.80039    validation_1-rmse:1504.43449
[953]   validation_0-rmse:1012.60096    validation_1-rmse:1504.65000
[954]   validation_0-rmse:1012.43213    validation_1-rmse:1504.64631
[955]   validation_0-rmse:1012.09444    validation_1-rmse:1504.65287
[956]   validation_0-rmse:1011.71483    validation_1-rmse:1504.70360
[957]   validation_0-rmse:1011.55434    validation_1-rmse:1504.68617
[958]   validation_0-rmse:1011.36260    validation_1-rmse:1504.76555
[959]   validation_0-rmse:1011.18467    validation_1-rmse:1504.80978
[960]   validation_0-rmse:1010.87470    validation_1-rmse:1504.66788
[961]   validation_0-rmse:1010.60379    validation_1-rmse:1504.80845
[962]   validation_0-rmse:1010.39873    validation_1-rmse:1504.96868
[963]   validation_0-rmse:1010.26097    validation_1-rmse:1505.17124
[964]   validation_0-rmse:1010.06809    validation_1-rmse:1505.01803
[965]   validation_0-rmse:1009.90732    validation_1-rmse:1504.97899
[966]   validation_0-rmse:1009.77361    validation_1-rmse:1504.94314
[967]   validation_0-rmse:1009.52933    validation_1-rmse:1504.81143
[968]   validation_0-rmse:1009.33270    validation_1-rmse:1504.98214
[969]   validation_0-rmse:1009.01365    validation_1-rmse:1505.21205
[970]   validation_0-rmse:1008.90402    validation_1-rmse:1505.18996
[971]   validation_0-rmse:1008.83899    validation_1-rmse:1505.22599
[972]   validation_0-rmse:1008.51821    validation_1-rmse:1505.14036
[973]   validation_0-rmse:1008.35194    validation_1-rmse:1505.31087
[974]   validation_0-rmse:1008.28483    validation_1-rmse:1505.18395
```

```
[975]   validation_0-rmse:1008.05890    validation_1-rmse:1505.12396
[976]   validation_0-rmse:1007.96144    validation_1-rmse:1505.19068
[977]   validation_0-rmse:1007.79064    validation_1-rmse:1505.20247
[978]   validation_0-rmse:1007.60101    validation_1-rmse:1505.27420
[979]   validation_0-rmse:1007.38756    validation_1-rmse:1505.28385
[980]   validation_0-rmse:1007.06481    validation_1-rmse:1505.21664
[981]   validation_0-rmse:1006.99796    validation_1-rmse:1505.34050
[982]   validation_0-rmse:1006.71624    validation_1-rmse:1505.41136
[983]   validation_0-rmse:1006.60157    validation_1-rmse:1505.44119
[984]   validation_0-rmse:1006.46090    validation_1-rmse:1505.66870
[985]   validation_0-rmse:1006.30868    validation_1-rmse:1505.49676
[986]   validation_0-rmse:1006.08696    validation_1-rmse:1505.47411
[987]   validation_0-rmse:1005.93874    validation_1-rmse:1505.40756
[988]   validation_0-rmse:1005.68408    validation_1-rmse:1505.40891
[989]   validation_0-rmse:1005.53526    validation_1-rmse:1505.47777
[990]   validation_0-rmse:1005.29572    validation_1-rmse:1505.50460
[991]   validation_0-rmse:1005.17072    validation_1-rmse:1505.51506
[992]   validation_0-rmse:1005.11431    validation_1-rmse:1505.53109
[993]   validation_0-rmse:1004.92751    validation_1-rmse:1505.62703
[994]   validation_0-rmse:1004.83161    validation_1-rmse:1505.68053
[995]   validation_0-rmse:1004.76457    validation_1-rmse:1505.80702
[996]   validation_0-rmse:1004.51827    validation_1-rmse:1505.74361
[997]   validation_0-rmse:1004.27164    validation_1-rmse:1505.62729
[998]   validation_0-rmse:1004.07170    validation_1-rmse:1505.63712
[999]   validation_0-rmse:1003.81519    validation_1-rmse:1505.54371
[1000]  validation_0-rmse:1003.72723    validation_1-rmse:1505.53217
[1001]  validation_0-rmse:1003.50914    validation_1-rmse:1505.66853
[1002]  validation_0-rmse:1003.22578    validation_1-rmse:1505.57449
[1003]  validation_0-rmse:1003.11831    validation_1-rmse:1505.49624
[1004]  validation_0-rmse:1003.00650    validation_1-rmse:1505.39673
[1005]  validation_0-rmse:1002.87407    validation_1-rmse:1505.62006
[1006]  validation_0-rmse:1002.73418    validation_1-rmse:1505.74930
[1007]  validation_0-rmse:1002.51078    validation_1-rmse:1505.85444
[1008]  validation_0-rmse:1002.27593    validation_1-rmse:1505.87004
[1009]  validation_0-rmse:1002.05930    validation_1-rmse:1505.83181
[1010]  validation_0-rmse:1001.79870    validation_1-rmse:1506.02103
[1011]  validation_0-rmse:1001.72985    validation_1-rmse:1506.03667
[1012]  validation_0-rmse:1001.58566    validation_1-rmse:1506.06280
[1013]  validation_0-rmse:1001.30205    validation_1-rmse:1506.01289
[1014]  validation_0-rmse:1001.14898    validation_1-rmse:1505.84165
[1015]  validation_0-rmse:1000.73884    validation_1-rmse:1505.70755
[1016]  validation_0-rmse:1000.57005    validation_1-rmse:1505.60758
[1017]  validation_0-rmse:1000.33221    validation_1-rmse:1505.68441
[1018]  validation_0-rmse:1000.17744    validation_1-rmse:1505.70991
[1019]  validation_0-rmse:1000.01850    validation_1-rmse:1505.81712
[1020]  validation_0-rmse:999.78646     validation_1-rmse:1505.87603
[1021]  validation_0-rmse:999.59964     validation_1-rmse:1505.81931
[1022]  validation_0-rmse:999.37573     validation_1-rmse:1505.86858
```

```
[1023]  validation_0-rmse:999.18095     validation_1-rmse:1506.02005
[1024]  validation_0-rmse:999.02381     validation_1-rmse:1505.89051
[1025]  validation_0-rmse:998.97534     validation_1-rmse:1505.93808
[1026]  validation_0-rmse:998.79887     validation_1-rmse:1506.08691
[1027]  validation_0-rmse:998.54379     validation_1-rmse:1506.02974
[1028]  validation_0-rmse:998.27784     validation_1-rmse:1505.98642
[1029]  validation_0-rmse:998.01820     validation_1-rmse:1505.93595
[1030]  validation_0-rmse:997.89690     validation_1-rmse:1505.94285
[1031]  validation_0-rmse:997.62515     validation_1-rmse:1505.90899
[1032]  validation_0-rmse:997.48157     validation_1-rmse:1505.88902
[1033]  validation_0-rmse:997.38346     validation_1-rmse:1506.02177
[1034]  validation_0-rmse:997.07119     validation_1-rmse:1506.07254
[1035]  validation_0-rmse:996.96806     validation_1-rmse:1506.13935
[1036]  validation_0-rmse:996.74464     validation_1-rmse:1506.42033
[1037]  validation_0-rmse:996.57733     validation_1-rmse:1506.45215
[1038]  validation_0-rmse:996.34591     validation_1-rmse:1506.36736
[1039]  validation_0-rmse:996.14425     validation_1-rmse:1506.49857
[1040]  validation_0-rmse:996.02333     validation_1-rmse:1506.47177
[1041]  validation_0-rmse:995.90990     validation_1-rmse:1506.46309
[1042]  validation_0-rmse:995.71174     validation_1-rmse:1506.65067
[1043]  validation_0-rmse:995.59116     validation_1-rmse:1506.62415
[1044]  validation_0-rmse:995.38553     validation_1-rmse:1506.74532
[1045]  validation_0-rmse:995.23073     validation_1-rmse:1506.61789
[1046]  validation_0-rmse:995.03324     validation_1-rmse:1506.57677
[1047]  validation_0-rmse:994.93933     validation_1-rmse:1506.63889
[1048]  validation_0-rmse:994.75662     validation_1-rmse:1506.78390
[1049]  validation_0-rmse:994.64978     validation_1-rmse:1506.77940
[1050]  validation_0-rmse:994.43466     validation_1-rmse:1506.74224
[1051]  validation_0-rmse:994.36753     validation_1-rmse:1506.74038
[1052]  validation_0-rmse:994.22171     validation_1-rmse:1506.68874
[1053]  validation_0-rmse:994.06687     validation_1-rmse:1506.54942
[1054]  validation_0-rmse:993.86071     validation_1-rmse:1506.45505
[1055]  validation_0-rmse:993.69564     validation_1-rmse:1506.42662
[1056]  validation_0-rmse:993.61727     validation_1-rmse:1506.37266
[1057]  validation_0-rmse:993.48054     validation_1-rmse:1506.56395
[1058]  validation_0-rmse:993.29930     validation_1-rmse:1506.51092
[1059]  validation_0-rmse:993.19167     validation_1-rmse:1506.31585
[1060]  validation_0-rmse:993.10933     validation_1-rmse:1506.43857
[1061]  validation_0-rmse:993.01942     validation_1-rmse:1506.46315
[1062]  validation_0-rmse:992.80057     validation_1-rmse:1506.53714
[1063]  validation_0-rmse:992.64269     validation_1-rmse:1506.59990
[1064]  validation_0-rmse:992.35448     validation_1-rmse:1506.64067
[1065]  validation_0-rmse:992.11191     validation_1-rmse:1506.62917
[1066]  validation_0-rmse:991.98477     validation_1-rmse:1506.66015
[1067]  validation_0-rmse:991.75329     validation_1-rmse:1506.44948
[1068]  validation_0-rmse:991.61681     validation_1-rmse:1506.54131
[1069]  validation_0-rmse:991.37601     validation_1-rmse:1506.49748
[1070]  validation_0-rmse:991.24456     validation_1-rmse:1506.64599
```

```
[1071]  validation_0-rmse:991.07177     validation_1-rmse:1506.66955
[1072]  validation_0-rmse:990.87170     validation_1-rmse:1506.63691
[1073]  validation_0-rmse:990.76361     validation_1-rmse:1506.71100
[1074]  validation_0-rmse:990.59886     validation_1-rmse:1506.65584
[1075]  validation_0-rmse:990.45530     validation_1-rmse:1506.62288
[1076]  validation_0-rmse:990.37315     validation_1-rmse:1506.62907
[1077]  validation_0-rmse:990.25018     validation_1-rmse:1506.60930
[1078]  validation_0-rmse:990.16416     validation_1-rmse:1506.69723
[1079]  validation_0-rmse:990.05949     validation_1-rmse:1506.60502
[1080]  validation_0-rmse:989.95673     validation_1-rmse:1506.78035
[1081]  validation_0-rmse:989.87849     validation_1-rmse:1506.89261
[1082]  validation_0-rmse:989.73683     validation_1-rmse:1507.04117
[1083]  validation_0-rmse:989.60729     validation_1-rmse:1507.13381
[1084]  validation_0-rmse:989.37018     validation_1-rmse:1507.21990
[1085]  validation_0-rmse:989.19753     validation_1-rmse:1507.33116
[1086]  validation_0-rmse:989.05816     validation_1-rmse:1507.41487
[1087]  validation_0-rmse:989.01623     validation_1-rmse:1507.33634
[1088]  validation_0-rmse:988.93251     validation_1-rmse:1507.28467
[1089]  validation_0-rmse:988.62390     validation_1-rmse:1507.33749
[1090]  validation_0-rmse:988.40250     validation_1-rmse:1507.33027
[1091]  validation_0-rmse:988.34139     validation_1-rmse:1507.49208
[1092]  validation_0-rmse:988.14585     validation_1-rmse:1507.45467
[1093]  validation_0-rmse:987.80335     validation_1-rmse:1507.40794
[1094]  validation_0-rmse:987.76901     validation_1-rmse:1507.35454
[1095]  validation_0-rmse:987.71377     validation_1-rmse:1507.40283
[1096]  validation_0-rmse:987.65216     validation_1-rmse:1507.38082
[1097]  validation_0-rmse:987.46492     validation_1-rmse:1507.32827
[1098]  validation_0-rmse:987.26159     validation_1-rmse:1507.41304
[1099]  validation_0-rmse:987.16363     validation_1-rmse:1507.34397
[1100]  validation_0-rmse:987.02467     validation_1-rmse:1507.33848
[1101]  validation_0-rmse:986.91076     validation_1-rmse:1507.45525
[1102]  validation_0-rmse:986.77805     validation_1-rmse:1507.58772
[1103]  validation_0-rmse:986.64301     validation_1-rmse:1507.50801
[1104]  validation_0-rmse:986.39756     validation_1-rmse:1507.59634
[1105]  validation_0-rmse:986.24222     validation_1-rmse:1507.73949
[1106]  validation_0-rmse:986.11193     validation_1-rmse:1507.82011
[1107]  validation_0-rmse:986.00118     validation_1-rmse:1507.84418
[1108]  validation_0-rmse:985.86345     validation_1-rmse:1507.97668
[1109]  validation_0-rmse:985.73359     validation_1-rmse:1508.05382
[1110]  validation_0-rmse:985.56881     validation_1-rmse:1508.10265
[1111]  validation_0-rmse:985.49316     validation_1-rmse:1508.02161
[1112]  validation_0-rmse:985.37004     validation_1-rmse:1508.06295
[1113]  validation_0-rmse:985.17154     validation_1-rmse:1508.03792
[1114]  validation_0-rmse:985.06883     validation_1-rmse:1507.95075
[1115]  validation_0-rmse:984.95092     validation_1-rmse:1508.04729
[1116]  validation_0-rmse:984.88024     validation_1-rmse:1508.06441
[1117]  validation_0-rmse:984.65292     validation_1-rmse:1507.94406
[1118]  validation_0-rmse:984.57811     validation_1-rmse:1508.05348
```

```
[1119]  validation_0-rmse:984.39997     validation_1-rmse:1508.11318
[1120]  validation_0-rmse:984.32392     validation_1-rmse:1507.97323
[1121]  validation_0-rmse:984.24174     validation_1-rmse:1507.90861
[1122]  validation_0-rmse:984.04066     validation_1-rmse:1507.84438
[1123]  validation_0-rmse:983.94688     validation_1-rmse:1507.80095
[1124]  validation_0-rmse:983.76974     validation_1-rmse:1507.91296
[1125]  validation_0-rmse:983.71743     validation_1-rmse:1507.89336
[1126]  validation_0-rmse:983.57050     validation_1-rmse:1507.97154
[1127]  validation_0-rmse:983.40713     validation_1-rmse:1507.97685
[1128]  validation_0-rmse:983.25408     validation_1-rmse:1508.03360
[1129]  validation_0-rmse:983.04600     validation_1-rmse:1508.01662
[1130]  validation_0-rmse:982.88805     validation_1-rmse:1508.11822
[1131]  validation_0-rmse:982.82899     validation_1-rmse:1508.21422
[1132]  validation_0-rmse:982.63729     validation_1-rmse:1508.20126
[1133]  validation_0-rmse:982.55591     validation_1-rmse:1508.29758
[1134]  validation_0-rmse:982.37133     validation_1-rmse:1508.30374
[1135]  validation_0-rmse:982.22874     validation_1-rmse:1508.26463
[1136]  validation_0-rmse:982.05116     validation_1-rmse:1508.38309
[1137]  validation_0-rmse:981.81441     validation_1-rmse:1508.26707
[1138]  validation_0-rmse:981.64177     validation_1-rmse:1508.31436
[1139]  validation_0-rmse:981.37748     validation_1-rmse:1508.31124
[1140]  validation_0-rmse:981.29507     validation_1-rmse:1508.26956
[1141]  validation_0-rmse:981.27099     validation_1-rmse:1508.22467
[1142]  validation_0-rmse:981.07656     validation_1-rmse:1508.18030
[1143]  validation_0-rmse:980.89684     validation_1-rmse:1508.33918
[1144]  validation_0-rmse:980.67859     validation_1-rmse:1508.32655
[1145]  validation_0-rmse:980.48040     validation_1-rmse:1508.22882
[1146]  validation_0-rmse:980.38813     validation_1-rmse:1508.53540
[1147]  validation_0-rmse:980.18556     validation_1-rmse:1508.49449
[1148]  validation_0-rmse:980.03776     validation_1-rmse:1508.54667
[1149]  validation_0-rmse:979.91686     validation_1-rmse:1508.59900
[1150]  validation_0-rmse:979.75711     validation_1-rmse:1508.46112
[1151]  validation_0-rmse:979.68789     validation_1-rmse:1508.45550
[1152]  validation_0-rmse:979.52610     validation_1-rmse:1508.45344
[1153]  validation_0-rmse:979.46809     validation_1-rmse:1508.37864
[1154]  validation_0-rmse:979.28882     validation_1-rmse:1508.34238
[1155]  validation_0-rmse:979.17769     validation_1-rmse:1508.31064
[1156]  validation_0-rmse:979.05144     validation_1-rmse:1508.33339
[1157]  validation_0-rmse:978.87302     validation_1-rmse:1508.35090
[1158]  validation_0-rmse:978.74157     validation_1-rmse:1508.30892
[1159]  validation_0-rmse:978.62151     validation_1-rmse:1508.18763
[1160]  validation_0-rmse:978.37719     validation_1-rmse:1508.16475
[1161]  validation_0-rmse:978.29220     validation_1-rmse:1508.18733
[1162]  validation_0-rmse:978.18435     validation_1-rmse:1508.24429
[1163]  validation_0-rmse:978.07777     validation_1-rmse:1508.44084
[1164]  validation_0-rmse:977.98413     validation_1-rmse:1508.37611
[1165]  validation_0-rmse:977.85687     validation_1-rmse:1508.51560
[1166]  validation_0-rmse:977.69409     validation_1-rmse:1508.42450
```

```
[1167]  validation_0-rmse:977.61540     validation_1-rmse:1508.47365
[1168]  validation_0-rmse:977.57400     validation_1-rmse:1508.53251
[1169]  validation_0-rmse:977.52260     validation_1-rmse:1508.52499
[1170]  validation_0-rmse:977.41279     validation_1-rmse:1508.68382
[1171]  validation_0-rmse:977.22690     validation_1-rmse:1508.59814
[1172]  validation_0-rmse:977.12294     validation_1-rmse:1508.54314
[1173]  validation_0-rmse:976.96830     validation_1-rmse:1508.52910
[1174]  validation_0-rmse:976.82662     validation_1-rmse:1508.55521
[1175]  validation_0-rmse:976.72009     validation_1-rmse:1508.53352
[1176]  validation_0-rmse:976.46305     validation_1-rmse:1508.46306
[1177]  validation_0-rmse:976.34277     validation_1-rmse:1508.49314
[1178]  validation_0-rmse:976.27037     validation_1-rmse:1508.42204
[1179]  validation_0-rmse:976.16264     validation_1-rmse:1508.60363
[1180]  validation_0-rmse:976.04829     validation_1-rmse:1508.52996
[1181]  validation_0-rmse:975.92596     validation_1-rmse:1508.48875
[1182]  validation_0-rmse:975.89236     validation_1-rmse:1508.41205
[1183]  validation_0-rmse:975.81010     validation_1-rmse:1508.47496
[1184]  validation_0-rmse:975.61703     validation_1-rmse:1508.39814
[1185]  validation_0-rmse:975.49546     validation_1-rmse:1508.59729
[1186]  validation_0-rmse:975.43579     validation_1-rmse:1508.49698
[1187]  validation_0-rmse:975.37731     validation_1-rmse:1508.57190
[1188]  validation_0-rmse:975.25119     validation_1-rmse:1508.61369
[1189]  validation_0-rmse:975.13802     validation_1-rmse:1508.74385
[1190]  validation_0-rmse:975.08332     validation_1-rmse:1508.65786
[1191]  validation_0-rmse:974.97736     validation_1-rmse:1508.60424
[1192]  validation_0-rmse:974.96780     validation_1-rmse:1508.57324
[1193]  validation_0-rmse:974.79213     validation_1-rmse:1508.55144
[1194]  validation_0-rmse:974.60555     validation_1-rmse:1508.67863
[1195]  validation_0-rmse:974.53755     validation_1-rmse:1508.73066
[1196]  validation_0-rmse:974.35621     validation_1-rmse:1508.61627
[1197]  validation_0-rmse:974.22236     validation_1-rmse:1508.49208
[1198]  validation_0-rmse:974.09642     validation_1-rmse:1508.41217
[1199]  validation_0-rmse:974.00603     validation_1-rmse:1508.34252
[1200]  validation_0-rmse:973.94795     validation_1-rmse:1508.30652
[1201]  validation_0-rmse:973.84787     validation_1-rmse:1508.40555
[1202]  validation_0-rmse:973.81659     validation_1-rmse:1508.29579
[1203]  validation_0-rmse:973.69231     validation_1-rmse:1508.51770
[1204]  validation_0-rmse:973.60213     validation_1-rmse:1508.48603
[1205]  validation_0-rmse:973.47134     validation_1-rmse:1508.56842
[1206]  validation_0-rmse:973.32364     validation_1-rmse:1508.53871
[1207]  validation_0-rmse:973.16277     validation_1-rmse:1508.63977
[1208]  validation_0-rmse:973.02635     validation_1-rmse:1508.72043
[1209]  validation_0-rmse:972.94082     validation_1-rmse:1508.72571
[1210]  validation_0-rmse:972.76761     validation_1-rmse:1508.74109
[1211]  validation_0-rmse:972.60101     validation_1-rmse:1508.71414
[1212]  validation_0-rmse:972.39645     validation_1-rmse:1508.87543
[1213]  validation_0-rmse:972.26715     validation_1-rmse:1508.81373
[1214]  validation_0-rmse:972.14161     validation_1-rmse:1508.78367
```

```
[1215]  validation_0-rmse:972.11111     validation_1-rmse:1508.90924
[1216]  validation_0-rmse:971.83038     validation_1-rmse:1508.85631
[1217]  validation_0-rmse:971.73638     validation_1-rmse:1508.77913
[1218]  validation_0-rmse:971.62291     validation_1-rmse:1508.77976
[1219]  validation_0-rmse:971.49190     validation_1-rmse:1508.80993
[1220]  validation_0-rmse:971.33894     validation_1-rmse:1508.93566
[1221]  validation_0-rmse:971.26620     validation_1-rmse:1509.01024
[1222]  validation_0-rmse:971.12442     validation_1-rmse:1509.04663
[1223]  validation_0-rmse:970.97384     validation_1-rmse:1509.04670
[1224]  validation_0-rmse:970.85501     validation_1-rmse:1509.16328
[1225]  validation_0-rmse:970.66435     validation_1-rmse:1509.12419
[1226]  validation_0-rmse:970.55735     validation_1-rmse:1509.20550
[1227]  validation_0-rmse:970.43654     validation_1-rmse:1509.28839
[1228]  validation_0-rmse:970.27892     validation_1-rmse:1509.28369
[1229]  validation_0-rmse:970.03543     validation_1-rmse:1509.23731
[1230]  validation_0-rmse:969.97891     validation_1-rmse:1509.08671
[1231]  validation_0-rmse:969.87803     validation_1-rmse:1509.18616
[1232]  validation_0-rmse:969.70152     validation_1-rmse:1509.12281
[1233]  validation_0-rmse:969.60219     validation_1-rmse:1509.19983
[1234]  validation_0-rmse:969.53801     validation_1-rmse:1509.29543
[1235]  validation_0-rmse:969.44688     validation_1-rmse:1509.29715
[1236]  validation_0-rmse:969.31021     validation_1-rmse:1509.25463
[1237]  validation_0-rmse:969.17950     validation_1-rmse:1509.35085
[1238]  validation_0-rmse:969.05509     validation_1-rmse:1509.51011
[1239]  validation_0-rmse:968.92680     validation_1-rmse:1509.62484
[1240]  validation_0-rmse:968.73747     validation_1-rmse:1509.74123
[1241]  validation_0-rmse:968.63293     validation_1-rmse:1509.85055
[1242]  validation_0-rmse:968.53968     validation_1-rmse:1509.83158
[1243]  validation_0-rmse:968.38040     validation_1-rmse:1509.81623
[1244]  validation_0-rmse:968.29838     validation_1-rmse:1509.87199
[1245]  validation_0-rmse:968.04797     validation_1-rmse:1509.89582
[1246]  validation_0-rmse:968.00902     validation_1-rmse:1509.74048
[1247]  validation_0-rmse:967.87686     validation_1-rmse:1509.75085
[1248]  validation_0-rmse:967.75583     validation_1-rmse:1509.77890
[1249]  validation_0-rmse:967.61267     validation_1-rmse:1509.81929
[1250]  validation_0-rmse:967.52872     validation_1-rmse:1509.78421
[1251]  validation_0-rmse:967.47541     validation_1-rmse:1509.83396
[1252]  validation_0-rmse:967.38419     validation_1-rmse:1509.84487
[1253]  validation_0-rmse:967.25766     validation_1-rmse:1509.96153
[1254]  validation_0-rmse:967.17749     validation_1-rmse:1509.98314
[1255]  validation_0-rmse:967.10367     validation_1-rmse:1509.95854
[1256]  validation_0-rmse:966.94149     validation_1-rmse:1509.81596
[1257]  validation_0-rmse:966.70644     validation_1-rmse:1509.88848
[1258]  validation_0-rmse:966.68279     validation_1-rmse:1509.93813
[1259]  validation_0-rmse:966.56074     validation_1-rmse:1510.01595
[1260]  validation_0-rmse:966.52072     validation_1-rmse:1510.09501
[1261]  validation_0-rmse:966.41676     validation_1-rmse:1510.05416
[1262]  validation_0-rmse:966.36084     validation_1-rmse:1510.08842
```

```
[1263]  validation_0-rmse:966.22147     validation_1-rmse:1510.21203
[1264]  validation_0-rmse:966.07118     validation_1-rmse:1510.25615
[1265]  validation_0-rmse:965.94998     validation_1-rmse:1510.35846
```

[ ]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=500,
             enable_categorical=False, eval_metric='rmse', feature_types=None,
             gamma=0.1, grow_policy=None, importance_type='gain',
             interaction_constraints=None, learning_rate=0.1, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=8, max_leaves=None,
             min_child_weight=1, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=5000, n_jobs=7,
             num_parallel_tree=None, random_state=999, …)
```

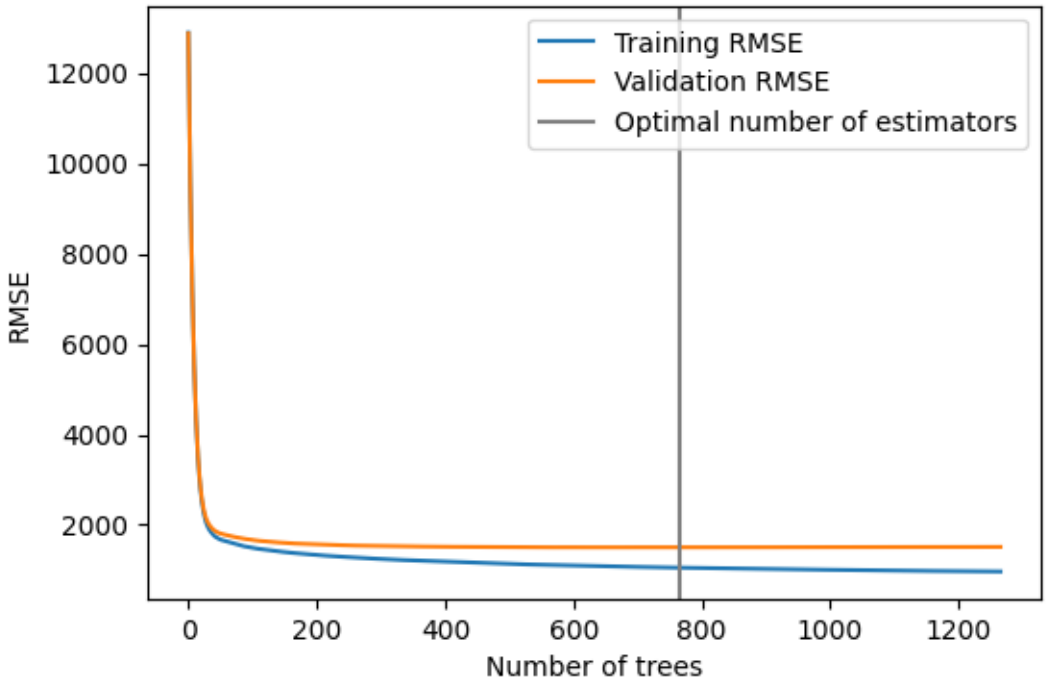
[ ]:
```
# Evaluate model performance, feature importance
xgb_init.best_score, xgb_init.feature_importances_
```

[ ]:
```
(1501.1759543651679,
 array([3.2010112e-02, 5.1271762e-03, 1.7771663e-01, 1.1765038e-02,
        9.9927038e-03, 2.6782090e-02, 4.5403946e-02, 9.7959675e-03,
        8.7768734e-03, 2.2518910e-03, 6.2637827e-03, 1.4571153e-02,
        0.0000000e+00, 2.9950289e-03, 9.7510880e-03, 2.4787907e-03,
        3.0240217e-02, 1.4371184e-03, 7.3296991e-03, 1.2987156e-01,
        3.7779097e-02, 0.0000000e+00, 7.0178308e-02, 0.0000000e+00,
        6.7099067e-04, 3.9648782e-03, 5.1681087e-03, 7.3949262e-03,
        8.1253209e-04, 1.4599781e-02, 3.2035496e-02, 2.9268703e-01,
        1.4802130e-04], dtype=float32))
```

[ ]:
```
# Visualize Evaluation of RMSE and optimal number of estimators for the given␣
 ↪learning rate
print(f'Optimal number of estimators: {xgb_init.best_iteration}'),
# Visualization
results = xgb_init.evals_result()
plt.figure(figsize=(6, 4))
plt.plot(results["validation_0"]["rmse"], label="Training RMSE")
plt.plot(results["validation_1"]["rmse"], label="Validation RMSE")
plt.axvline(xgb_init.best_iteration, color="gray", label="Optimal number of␣
 ↪estimators")
plt.xlabel("Number of trees")
plt.ylabel("RMSE")
plt.legend()
```

```
Optimal number of estimators: 765
```

[ ]:
```
<matplotlib.legend.Legend at 0x7c8062f57b20>
```



[ ]:
```
# Predict test data set to have benchmark for model performance
y_pred_init = xgb_init.predict(X_test)
rmse_init = np.sqrt(mean_squared_error(y_test, y_pred_init))
r2_init = r2_score(y_test, y_pred_init)

rmse_init, r2_init
```

[ ]:
```
(1454.6292530450798, 0.9768037829316646)
```

For the given learning rate of 0.1, our initial XGBoost model estimates 765 trees to obtain its best predictions. The final model performance, measured by evaluating predictions on the test sample, shows an asbolute mean error of 1454.62 and an R2 score of 0.9781. Thus, the default hyperparameter specifications of our first model already provide very good predicitons which is in line with the overall high predicition power of the XGBoost algorithm. Especially the R2 score indicates that our baseline model explains most of the variance (97.68%) of our target variable.

B) Model Tuning Step 1: Tuning Tree Depth and Child Weight

To improve our baseline model, we start by tuning the hyperparameters of max_depth and min_child_weight as they have the biggest impact on growing the xgbtrees.

[ ]:
```
# Define xgb_grid model for parameter tuning with n_estimators = 624 and␣
↪without early_stopping_rounds
xgb_grid = xgb.XGBRegressor(objective = "reg:squarederror", # Objective␣
↪function to be minimized by XGBoost model
```

```
                            eval_metric="rmse",
                            base_score = 0.5, # Initial prediction score for␣
↪all observations (default value).
                            booster = "gbtree", # Boosting type, here gradient␣
↪boosting tree
                            learning_rate = 0.1, # Learning rate of XGBoost␣
↪model determining the speed of convergence (default value)
                            n_estimators = 765, # Optimal number of estimators␣
↪for given learning rate of 0.1
                            max_depth = 8, # Maximum tree depth (to be tuned)
                            min_child_weight = 1, # Minimum number of instances␣
↪needed per child (to be tuned)
                            reg_lambda = 1,  # L2 regularization parameter␣
↪scaling the similiratiy scores and controlling for nodes containing only few␣
↪parameters (to be tuned)
                            gamma = 0.1, # Similiarity score threshold (pruning␣
↪parameter) to create new split (to be tuned)
                            subsample = 0.8, # Sample size of training data to␣
↪be used at each iteration (to be tuned)
                            sampling_method = "uniform", # Determines␣
↪distribution from which subsample is randomly drawn.
                            importance_type = "gain", # Criteria for the␣
↪model's feature_importance property
                            n_jobs = 7, # Number of CPU kernels to be used for␣
↪model training (time reduction)
                            verbosity = 2,
                            random_state = 999
                            )
```

[ ]:
```
# Define first parameter grid for max_depth and min_child_weight
param_grid1 = {"max_depth" : [2, 4, 6, 12],
               "min_child_weight" : [1, 2, 6, 10]
               }

# Define Grid search model
xgb1 = GridSearchCV(estimator = xgb_grid,
                    param_grid = param_grid1,
                    scoring = ["neg_root_mean_squared_error", "r2"],
                    refit = "neg_root_mean_squared_error", # Define metric␣
↪according to which the model's parameters will be set after the grid search␣
↪(to directly use xgb1 for predicting)
                    cv = 4,
                    verbose = 3,
                    return_train_score = True, # Set to true such that the␣
↪cv_results will also include the training error scores
                    )
```

```
# Fit model
xgb1.fit(X_train, y_train)
```

```
Fitting 4 folds for each of 16 candidates, totalling 64 fits
[CV 1/4] END max_depth=2, min_child_weight=1; neg_root_mean_squared_error:
(train=-2186.644, test=-2206.411) r2: (train=0.947, test=0.946) total time=
5.3s
[CV 2/4] END max_depth=2, min_child_weight=1; neg_root_mean_squared_error:
(train=-2197.699, test=-2239.791) r2: (train=0.946, test=0.945) total time=
4.8s
[CV 3/4] END max_depth=2, min_child_weight=1; neg_root_mean_squared_error:
(train=-2198.202, test=-2222.022) r2: (train=0.947, test=0.945) total time=
5.4s
[CV 4/4] END max_depth=2, min_child_weight=1; neg_root_mean_squared_error:
(train=-2190.028, test=-2196.274) r2: (train=0.947, test=0.946) total time=
4.7s
[CV 1/4] END max_depth=2, min_child_weight=2; neg_root_mean_squared_error:
(train=-2188.307, test=-2212.578) r2: (train=0.947, test=0.946) total time=
5.5s
[CV 2/4] END max_depth=2, min_child_weight=2; neg_root_mean_squared_error:
(train=-2203.050, test=-2246.620) r2: (train=0.946, test=0.944) total time=
4.7s
[CV 3/4] END max_depth=2, min_child_weight=2; neg_root_mean_squared_error:
(train=-2198.016, test=-2221.121) r2: (train=0.947, test=0.945) total time=
5.4s
[CV 4/4] END max_depth=2, min_child_weight=2; neg_root_mean_squared_error:
(train=-2181.110, test=-2194.551) r2: (train=0.947, test=0.946) total time=
4.7s
[CV 1/4] END max_depth=2, min_child_weight=6; neg_root_mean_squared_error:
(train=-2187.980, test=-2213.108) r2: (train=0.947, test=0.946) total time=
4.9s
[CV 2/4] END max_depth=2, min_child_weight=6; neg_root_mean_squared_error:
(train=-2200.299, test=-2238.124) r2: (train=0.946, test=0.945) total time=
5.1s
[CV 3/4] END max_depth=2, min_child_weight=6; neg_root_mean_squared_error:
(train=-2196.230, test=-2215.164) r2: (train=0.947, test=0.945) total time=
4.8s
[CV 4/4] END max_depth=2, min_child_weight=6; neg_root_mean_squared_error:
(train=-2183.882, test=-2196.984) r2: (train=0.947, test=0.946) total time=
5.5s
[CV 1/4] END max_depth=2, min_child_weight=10; neg_root_mean_squared_error:
(train=-2184.584, test=-2209.437) r2: (train=0.947, test=0.946) total time=
4.7s
[CV 2/4] END max_depth=2, min_child_weight=10; neg_root_mean_squared_error:
(train=-2201.674, test=-2238.561) r2: (train=0.946, test=0.945) total time=
5.4s
[CV 3/4] END max_depth=2, min_child_weight=10; neg_root_mean_squared_error:
```

```
(train=-2196.300, test=-2214.495) r2: (train=0.947, test=0.945) total time=
4.7s
[CV 4/4] END max_depth=2, min_child_weight=10; neg_root_mean_squared_error:
(train=-2182.940, test=-2197.607) r2: (train=0.947, test=0.946) total time=
5.4s
[CV 1/4] END max_depth=4, min_child_weight=1; neg_root_mean_squared_error:
(train=-1598.344, test=-1684.412) r2: (train=0.972, test=0.969) total time=
6.6s
[CV 2/4] END max_depth=4, min_child_weight=1; neg_root_mean_squared_error:
(train=-1599.448, test=-1673.478) r2: (train=0.972, test=0.969) total time=
7.3s
[CV 3/4] END max_depth=4, min_child_weight=1; neg_root_mean_squared_error:
(train=-1600.287, test=-1670.386) r2: (train=0.972, test=0.969) total time=
7.0s
[CV 4/4] END max_depth=4, min_child_weight=1; neg_root_mean_squared_error:
(train=-1600.793, test=-1675.956) r2: (train=0.972, test=0.969) total time=
6.5s
[CV 1/4] END max_depth=4, min_child_weight=2; neg_root_mean_squared_error:
(train=-1597.823, test=-1683.012) r2: (train=0.972, test=0.969) total time=
7.3s
[CV 2/4] END max_depth=4, min_child_weight=2; neg_root_mean_squared_error:
(train=-1600.638, test=-1676.504) r2: (train=0.972, test=0.969) total time=
7.5s
[CV 3/4] END max_depth=4, min_child_weight=2; neg_root_mean_squared_error:
(train=-1604.225, test=-1670.716) r2: (train=0.972, test=0.969) total time=
8.1s
[CV 4/4] END max_depth=4, min_child_weight=2; neg_root_mean_squared_error:
(train=-1599.033, test=-1678.539) r2: (train=0.972, test=0.969) total time=
9.2s
[CV 1/4] END max_depth=4, min_child_weight=6; neg_root_mean_squared_error:
(train=-1602.451, test=-1688.454) r2: (train=0.971, test=0.969) total time=
11.2s
[CV 2/4] END max_depth=4, min_child_weight=6; neg_root_mean_squared_error:
(train=-1604.915, test=-1677.421) r2: (train=0.971, test=0.969) total time=
10.1s
[CV 3/4] END max_depth=4, min_child_weight=6; neg_root_mean_squared_error:
(train=-1605.149, test=-1673.428) r2: (train=0.972, test=0.969) total time=
7.4s
[CV 4/4] END max_depth=4, min_child_weight=6; neg_root_mean_squared_error:
(train=-1602.721, test=-1682.577) r2: (train=0.972, test=0.968) total time=
6.5s
[CV 1/4] END max_depth=4, min_child_weight=10; neg_root_mean_squared_error:
(train=-1609.642, test=-1694.797) r2: (train=0.971, test=0.968) total time=
7.3s
[CV 2/4] END max_depth=4, min_child_weight=10; neg_root_mean_squared_error:
(train=-1610.442, test=-1681.634) r2: (train=0.971, test=0.969) total time=
12.2s
[CV 3/4] END max_depth=4, min_child_weight=10; neg_root_mean_squared_error:
```

```
(train=-1608.684, test=-1675.150) r2: (train=0.971, test=0.969) total time=
10.7s
[CV 4/4] END max_depth=4, min_child_weight=10; neg_root_mean_squared_error:
(train=-1605.964, test=-1684.973) r2: (train=0.971, test=0.968) total time=
9.6s
[CV 1/4] END max_depth=6, min_child_weight=1; neg_root_mean_squared_error:
(train=-1289.285, test=-1520.083) r2: (train=0.982, test=0.975) total time=
13.1s
[CV 2/4] END max_depth=6, min_child_weight=1; neg_root_mean_squared_error:
(train=-1288.508, test=-1506.737) r2: (train=0.982, test=0.975) total time=
13.2s
[CV 3/4] END max_depth=6, min_child_weight=1; neg_root_mean_squared_error:
(train=-1285.298, test=-1504.553) r2: (train=0.982, test=0.975) total time=
9.5s
[CV 4/4] END max_depth=6, min_child_weight=1; neg_root_mean_squared_error:
(train=-1286.656, test=-1505.362) r2: (train=0.982, test=0.975) total time=
9.6s
[CV 1/4] END max_depth=6, min_child_weight=2; neg_root_mean_squared_error:
(train=-1291.126, test=-1514.695) r2: (train=0.981, test=0.975) total time=
9.6s
[CV 2/4] END max_depth=6, min_child_weight=2; neg_root_mean_squared_error:
(train=-1291.129, test=-1504.336) r2: (train=0.982, test=0.975) total time=
9.5s
[CV 3/4] END max_depth=6, min_child_weight=2; neg_root_mean_squared_error:
(train=-1289.613, test=-1505.174) r2: (train=0.982, test=0.975) total time=
10.5s
[CV 4/4] END max_depth=6, min_child_weight=2; neg_root_mean_squared_error:
(train=-1288.307, test=-1507.109) r2: (train=0.982, test=0.975) total time=
12.8s
[CV 1/4] END max_depth=6, min_child_weight=6; neg_root_mean_squared_error:
(train=-1310.230, test=-1516.495) r2: (train=0.981, test=0.975) total time=
14.3s
[CV 2/4] END max_depth=6, min_child_weight=6; neg_root_mean_squared_error:
(train=-1310.950, test=-1505.617) r2: (train=0.981, test=0.975) total time=
12.1s
[CV 3/4] END max_depth=6, min_child_weight=6; neg_root_mean_squared_error:
(train=-1306.202, test=-1513.876) r2: (train=0.981, test=0.974) total time=
14.0s
[CV 4/4] END max_depth=6, min_child_weight=6; neg_root_mean_squared_error:
(train=-1309.743, test=-1508.060) r2: (train=0.981, test=0.975) total time=
11.9s
[CV 1/4] END max_depth=6, min_child_weight=10; neg_root_mean_squared_error:
(train=-1323.983, test=-1520.893) r2: (train=0.981, test=0.975) total time=
9.5s
[CV 2/4] END max_depth=6, min_child_weight=10; neg_root_mean_squared_error:
(train=-1326.943, test=-1510.893) r2: (train=0.980, test=0.975) total time=
9.5s
[CV 3/4] END max_depth=6, min_child_weight=10; neg_root_mean_squared_error:
```

```
(train=-1326.077, test=-1511.840) r2: (train=0.981, test=0.975) total time=
9.4s
[CV 4/4] END max_depth=6, min_child_weight=10; neg_root_mean_squared_error:
(train=-1325.951, test=-1521.189) r2: (train=0.981, test=0.974) total time=
8.8s
[CV 1/4] END max_depth=12, min_child_weight=1; neg_root_mean_squared_error:
(train=-716.091, test=-1616.809) r2: (train=0.994, test=0.971) total time=
46.1s
[CV 2/4] END max_depth=12, min_child_weight=1; neg_root_mean_squared_error:
(train=-736.909, test=-1587.119) r2: (train=0.994, test=0.972) total time=
33.7s
[CV 3/4] END max_depth=12, min_child_weight=1; neg_root_mean_squared_error:
(train=-725.631, test=-1576.528) r2: (train=0.994, test=0.972) total time=
34.1s
[CV 4/4] END max_depth=12, min_child_weight=1; neg_root_mean_squared_error:
(train=-724.571, test=-1605.517) r2: (train=0.994, test=0.971) total time=
34.2s
[CV 1/4] END max_depth=12, min_child_weight=2; neg_root_mean_squared_error:
(train=-731.350, test=-1611.321) r2: (train=0.994, test=0.971) total time=
31.4s
[CV 2/4] END max_depth=12, min_child_weight=2; neg_root_mean_squared_error:
(train=-750.289, test=-1588.567) r2: (train=0.994, test=0.972) total time=
31.0s
[CV 3/4] END max_depth=12, min_child_weight=2; neg_root_mean_squared_error:
(train=-739.857, test=-1580.452) r2: (train=0.994, test=0.972) total time=
30.1s
[CV 4/4] END max_depth=12, min_child_weight=2; neg_root_mean_squared_error:
(train=-736.967, test=-1600.951) r2: (train=0.994, test=0.971) total time=
30.3s
[CV 1/4] END max_depth=12, min_child_weight=6; neg_root_mean_squared_error:
(train=-797.539, test=-1576.633) r2: (train=0.993, test=0.973) total time=
24.7s
[CV 2/4] END max_depth=12, min_child_weight=6; neg_root_mean_squared_error:
(train=-808.340, test=-1569.685) r2: (train=0.993, test=0.973) total time=
25.5s
[CV 3/4] END max_depth=12, min_child_weight=6; neg_root_mean_squared_error:
(train=-802.802, test=-1560.691) r2: (train=0.993, test=0.973) total time=
25.6s
[CV 4/4] END max_depth=12, min_child_weight=6; neg_root_mean_squared_error:
(train=-801.810, test=-1582.247) r2: (train=0.993, test=0.972) total time=
24.9s
[CV 1/4] END max_depth=12, min_child_weight=10; neg_root_mean_squared_error:
(train=-857.953, test=-1563.978) r2: (train=0.992, test=0.973) total time=
21.8s
[CV 2/4] END max_depth=12, min_child_weight=10; neg_root_mean_squared_error:
(train=-862.994, test=-1546.689) r2: (train=0.992, test=0.974) total time=
22.7s
[CV 3/4] END max_depth=12, min_child_weight=10; neg_root_mean_squared_error:
```

```
(train=-859.355, test=-1543.033) r2: (train=0.992, test=0.974) total time=
22.6s
[CV 4/4] END max_depth=12, min_child_weight=10; neg_root_mean_squared_error:
(train=-859.951, test=-1559.240) r2: (train=0.992, test=0.973) total time=
23.1s
```

[ ]:
```
GridSearchCV(cv=4,
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    callbacks=None, colsample_bylevel=None,
                                    colsample_bynode=None,
                                    colsample_bytree=None, device=None,
                                    early_stopping_rounds=None,
                                    enable_categorical=False,
                                    eval_metric='rmse', feature_types=None,
                                    gamma=0.1, grow_policy=None,
                                    importance_type='gain',
                                    interaction_constraints=None,
                                    learning_rate=…
                                    max_depth=8, max_leaves=None,
                                    min_child_weight=1, missing=nan,
                                    monotone_constraints=None,
                                    multi_strategy=None, n_estimators=765,
                                    n_jobs=7, num_parallel_tree=None,
                                    random_state=999, …),
             param_grid={'max_depth': [2, 4, 6, 12],
                         'min_child_weight': [1, 2, 6, 10]},
             refit='neg_root_mean_squared_error', return_train_score=True,
             scoring=['neg_root_mean_squared_error', 'r2'], verbose=3)
```

[ ]:
```
xgb1.best_params_, xgb1.best_score_
```

[ ]:
```
({'max_depth': 6, 'min_child_weight': 2}, -1507.8284062380908)
```

[ ]:
```
y_pred1 = xgb1.predict(X_test)
rmse_1 = np.sqrt(mean_squared_error(y_test ,y_pred1))
r2_1 = r2_score(y_test, y_pred1)

rmse_1, r2_1
```

[ ]:
```
(1466.7774450217973, 0.9764147232769508)
```

Step 1.1: Serach for optimal Tree Depth

In a next step, we search for potentially better values for the tree depth and child weight, by searching around the best parameters from the previous step.

[ ]:
```
# Define first parameter grid for max_depth and min_child_weight
param_grid1_1 = {"max_depth" : [5, 6, 7],
                 "min_child_weight" : [1 ,2]
```

```
              }

# Define Grid search model
xgb1_1 = GridSearchCV(estimator = xgb_grid,
                      param_grid = param_grid1_1,
                      scoring = ["neg_root_mean_squared_error", "r2"],
                      refit = "neg_root_mean_squared_error", # Define metric␣
 ↪according to which the model's parameters will be set after the grid search␣
 ↪(to directly use xgb1 for predicting)
                      cv = 4,
                      verbose = 3,
                      return_train_score = True, # Set to true such that the␣
 ↪cv_results will also include the training error scores
                      )

# Fit model
xgb1_1.fit(X_train, y_train)
```

```
Fitting 4 folds for each of 1 candidates, totalling 4 fits
[CV 1/4] END max_depth=7, min_child_weight=1; neg_root_mean_squared_error:
(train=-1154.122, test=-1495.410) r2: (train=0.985, test=0.975) total time=
28.6s
[CV 2/4] END max_depth=7, min_child_weight=1; neg_root_mean_squared_error:
(train=-1156.008, test=-1478.452) r2: (train=0.985, test=0.976) total time=
20.7s
[CV 3/4] END max_depth=7, min_child_weight=1; neg_root_mean_squared_error:
(train=-1154.527, test=-1481.698) r2: (train=0.985, test=0.976) total time=
19.9s
[CV 4/4] END max_depth=7, min_child_weight=1; neg_root_mean_squared_error:
(train=-1150.786, test=-1484.296) r2: (train=0.985, test=0.975) total time=
20.2s
```

[ ]:
```
GridSearchCV(cv=4,
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    callbacks=None, colsample_bylevel=None,
                                    colsample_bynode=None,
                                    colsample_bytree=None, device=None,
                                    early_stopping_rounds=None,
                                    enable_categorical=False,
                                    eval_metric='rmse', feature_types=None,
                                    gamma=0.1, grow_policy=None,
                                    importance_type='gain',
                                    interaction_constraints=None,
                                    learning_rate=…
                                    max_cat_to_onehot=None, max_delta_step=None,
                                    max_depth=8, max_leaves=None,
                                    min_child_weight=1, missing=nan,
```

```
                                monotone_constraints=None,
                                multi_strategy=None, n_estimators=765,
                                n_jobs=7, num_parallel_tree=None,
                                random_state=999, …),
             param_grid={'max_depth': [7], 'min_child_weight': [1]},
             refit='neg_root_mean_squared_error', return_train_score=True,
             scoring=['neg_root_mean_squared_error', 'r2'], verbose=3)
```

[ ]:
```
xgb1_1.best_params_, xgb1_1.best_score_
```

[ ]:
```
({'max_depth': 7, 'min_child_weight': 1}, -1484.9641657794384)
```

[ ]:
```
y_pred1_1 = xgb1_1.predict(X_test)
rmse_1_1 = np.sqrt(mean_squared_error(y_test ,y_pred1_1))
r2_1_1 = r2_score(y_test, y_pred1_1)

rmse_1_1, r2_1_1
```

[ ]:
```
(1432.9995767548828, 0.9774884885932376)
```

We can see, that for the given learning rate and number of estimators, the optimal tree depth is 7 and the the optimal minimum child weight is 1. Using these paraemeters, we can observe that our error metric slightly decreased by approx. 30 points. We can also observe a slight improvement in prediction accuracy.

Step 2: Tuning Gamma

[ ]:
```
# Define second parameter grid
param_grid2 = {"gamma" : [0, 00.1, 0.1, 1, 10, 100]

}

# Define Grid search model
# Define Grid search model
xgb2 = GridSearchCV(estimator = xgb1_1.best_estimator_,
                    param_grid = param_grid2,
                    scoring = ["neg_root_mean_squared_error", "r2"],
                    refit = "neg_root_mean_squared_error", # Define metric␣
 ↪according to which the model's parameters will be set after the grid search␣
 ↪(to directly use xgb1 for predicting)
                    cv = 4,
                    verbose = 3,
                    return_train_score = True, # Set to true such that the␣
 ↪cv_results will also include the training error scores
                    )

# Fit model
xgb2.fit(X_train, y_train)
```

```
Fitting 4 folds for each of 1 candidates, totalling 4 fits
[CV 1/4] END gamma=0; neg_root_mean_squared_error: (train=-1154.122,
test=-1495.410) r2: (train=0.985, test=0.975) total time=  26.5s
[CV 2/4] END gamma=0; neg_root_mean_squared_error: (train=-1156.008,
test=-1478.452) r2: (train=0.985, test=0.976) total time=  19.8s
[CV 3/4] END gamma=0; neg_root_mean_squared_error: (train=-1154.527,
test=-1481.698) r2: (train=0.985, test=0.976) total time=  21.2s
[CV 4/4] END gamma=0; neg_root_mean_squared_error: (train=-1150.786,
test=-1484.296) r2: (train=0.985, test=0.975) total time=  19.8s
```

[ ]:
```
GridSearchCV(cv=4,
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    callbacks=None, colsample_bylevel=None,
                                    colsample_bynode=None,
                                    colsample_bytree=None, device=None,
                                    early_stopping_rounds=None,
                                    enable_categorical=False,
                                    eval_metric='rmse', feature_types=None,
                                    gamma=0.1, grow_policy=None,
                                    importance_type='gain',
                                    interaction_constraints=None,
                                    learning_rate=…
                                    max_cat_to_onehot=None, max_delta_step=None,
                                    max_depth=7, max_leaves=None,
                                    min_child_weight=1, missing=nan,
                                    monotone_constraints=None,
                                    multi_strategy=None, n_estimators=765,
                                    n_jobs=7, num_parallel_tree=None,
                                    random_state=999, …),
             param_grid={'gamma': [0]}, refit='neg_root_mean_squared_error',
             return_train_score=True,
             scoring=['neg_root_mean_squared_error', 'r2'], verbose=3)
```

[ ]:
```
xgb2.best_params_, xgb2.best_score_
```

[ ]:
```
({'gamma': 0}, -1484.9641657794384)
```

[ ]:
```
y_pred2 = xgb2.predict(X_test)
rmse_2 = np.sqrt(mean_squared_error(y_test ,y_pred2))
r2_2 = r2_score(y_test, y_pred2)

rmse_2, r2_2
```

[ ]:
```
(1432.9995767548828, 0.9774884885932376)
```

We can observe that our initial gamma value of 0.1 did not enhance the model performance. Setting the scoring threshold to create a new child node to 0 yields the same predictive power. This is surprising, since the child weight is set to a quite low value and thus model overfitting should be expected. However, cross validating over values on child weight and gamma suggest more

complex trees. An explanation herefore, might be that the tree depth regulates the tree complexity sufficiently.

Step 3: Tuning Subsample Size

[ ]:
```
# Define third parameter grid
param_grid3 = {"subsample" : [0.5, 0.6, 0.7, 0.8, 0.9, 1]
               }

# Define Grid search model
xgb3 = GridSearchCV(estimator = xgb2.best_estimator_,
                    param_grid = param_grid3,
                    scoring = ["neg_root_mean_squared_error", "r2"],
                    refit = "neg_root_mean_squared_error", # Define metric␣
 ↪according to which the model's parameters will be set after the grid search␣
 ↪(to directly use xgb1 for predicting)
                    cv = 4,
                    verbose = 3,
                    return_train_score = True, # Set to true such that the␣
 ↪cv_results will also include the training error scores
                    )

# Fit model
xgb3.fit(X_train, y_train)
```

```
Fitting 4 folds for each of 1 candidates, totalling 4 fits
[CV 1/4] END subsample=1; neg_root_mean_squared_error: (train=-1167.886,
test=-1483.390) r2: (train=0.985, test=0.976) total time=  22.4s
[CV 2/4] END subsample=1; neg_root_mean_squared_error: (train=-1167.779,
test=-1472.158) r2: (train=0.985, test=0.976) total time=  17.8s
[CV 3/4] END subsample=1; neg_root_mean_squared_error: (train=-1165.212,
test=-1468.305) r2: (train=0.985, test=0.976) total time=  17.7s
[CV 4/4] END subsample=1; neg_root_mean_squared_error: (train=-1165.539,
test=-1478.739) r2: (train=0.985, test=0.976) total time=  22.2s
```

[ ]:
```
GridSearchCV(cv=4,
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    callbacks=None, colsample_bylevel=None,
                                    colsample_bynode=None,
                                    colsample_bytree=None, device=None,
                                    early_stopping_rounds=None,
                                    enable_categorical=False,
                                    eval_metric='rmse', feature_types=None,
                                    gamma=0, grow_policy=None,
                                    importance_type='gain',
                                    interaction_constraints=None,
                                    learning_rate=0…
                                    max_cat_to_onehot=None, max_delta_step=None,
```

```
                                    max_depth=7, max_leaves=None,
                                    min_child_weight=1, missing=nan,
                                    monotone_constraints=None,
                                    multi_strategy=None, n_estimators=765,
                                    n_jobs=7, num_parallel_tree=None,
                                    random_state=999, …),
             param_grid={'subsample': [1]}, refit='neg_root_mean_squared_error',
             return_train_score=True,
             scoring=['neg_root_mean_squared_error', 'r2'], verbose=3)
```

[ ]:
```
xgb3.best_params_, xgb3.best_score_
```

[ ]:
```
({'subsample': 1}, -1475.6477322792355)
```

[ ]:
```
y_pred3 = xgb3.predict(X_test)
rmse_3 = np.sqrt(mean_squared_error(y_test, y_pred3))
r2_3 = r2_score(y_test, y_pred3)

rmse_3, r2_3
```

[ ]:
```
(1424.5205748556557, 0.9777540999035317)
```

Step 4: Tuning L2 Regularization Parameter Lambda

[ ]:
```
# Define fourth parameter grid
param_grid4 = {"reg_lambda" : [0, 1, 2, 3, 4, 5, 10, 20, 40, 80, 120, 160]
               }

# Define Grid search model
xgb4 = GridSearchCV(estimator = xgb3.best_estimator_,
                    param_grid = param_grid4,
                    scoring = ["neg_root_mean_squared_error", "r2"],
                    refit = "neg_root_mean_squared_error",
                    n_jobs = 7, # Define number of CPU cores to use on your␣
 ↪machine
                    cv = 4,
                    verbose = 3,
                    return_train_score = True, # Set to true such that the␣
 ↪cv_results will also include the training error scores
                    )

# Fit model
xgb4.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits
```

[ ]:
```
GridSearchCV(cv=3,
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
```

```
                                callbacks=None, colsample_bylevel=None,
                                colsample_bynode=None,
                                colsample_bytree=None, device=None,
                                early_stopping_rounds=None,
                                enable_categorical=False,
                                eval_metric='rmse', feature_types=None,
                                gamma=0, grow_policy=None,
                                importance_type='gain',
                                interaction_constraints=None,
                                learning_rate=0…
                                max_depth=7, max_leaves=None,
                                min_child_weight=1, missing=nan,
                                monotone_constraints=None,
                                multi_strategy=None, n_estimators=765,
                                n_jobs=7, num_parallel_tree=None,
                                random_state=999, …),
             n_jobs=7,
             param_grid={'reg_lambda': [0, 1, 2, 3, 4, 5, 10, 20, 40, 80, 120,
                                        160]},
             refit='neg_root_mean_squared_error', return_train_score=True,
             scoring=['neg_root_mean_squared_error', 'r2'], verbose=3)
```

[ ]:
```
xgb4.best_params_, xgb4.best_score_
```

[ ]:
```
({'reg_lambda': 2}, -1487.870228739995)
```

[ ]:
```
y_pred4 = xgb4.predict(X_test)
rmse_4 = np.sqrt(mean_squared_error(y_test, y_pred4))
r2_4 = r2_score(y_test, y_pred4)

rmse_4, r2_4
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-25-f8d48aa9d089> in <cell line: 1>()
----> 1 y_pred4 = xgb4.predict(X_test)
      2 rmse_4 = np.sqrt(mean_squared_error(y_test, y_pred4))
      3 r2_4 = r2_score(y_test, y_pred4)
      4
      5 rmse_4, r2_4

NameError: name 'xgb4' is not defined
```

As we can see, increasing the L2 regularization parameter lamda slightly improves our model performance. To obtain the optimal value for lambda, we conduct a more detailed grid search around the value of 80.

Step 4.1: Searching for best L2 Regularization Parameter Lambda

[ ]:
```
# Define fourth parameter grid
param_grid4_1 = {"reg_lambda" : [1.5, 1.75, 2, 2.25, 2.5]
               }

# Define Grid search model
xgb4_1 = GridSearchCV(estimator = xgb3.best_estimator_,
                      param_grid = param_grid4_1,
                      scoring = ["neg_root_mean_squared_error", "r2"],
                      refit = "neg_root_mean_squared_error",
                      n_jobs = 7, # Define number of CPU cores to use on your␣
 ↪machine
                      cv = 4,
                      verbose = 3,
                      return_train_score = True, # Set to true such that the␣
 ↪cv_results will also include the training error scores
                      )

# Fit model
xgb4_1.fit(X_train, y_train)
```

```
Fitting 4 folds for each of 1 candidates, totalling 4 fits
```

[ ]:
```
GridSearchCV(cv=4,
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    callbacks=None, colsample_bylevel=None,
                                    colsample_bynode=None,
                                    colsample_bytree=None, device=None,
                                    early_stopping_rounds=None,
                                    enable_categorical=False,
                                    eval_metric='rmse', feature_types=None,
                                    gamma=0, grow_policy=None,
                                    importance_type='gain',
                                    interaction_constraints=None,
                                    learning_rate=0…
                                    max_cat_to_onehot=None, max_delta_step=None,
                                    max_depth=7, max_leaves=None,
                                    min_child_weight=1, missing=nan,
                                    monotone_constraints=None,
                                    multi_strategy=None, n_estimators=765,
                                    n_jobs=7, num_parallel_tree=None,
                                    random_state=999, …),
             n_jobs=7, param_grid={'reg_lambda': [1.5]},
             refit='neg_root_mean_squared_error', return_train_score=True,
             scoring=['neg_root_mean_squared_error', 'r2'], verbose=3)
```

[ ]:
```
xgb4_1.best_params_, xgb4_1.best_score_
```

[ ]:
```
({'reg_lambda': 1.5}, -1475.5092485213968)
```

[ ]:
```
y_pred4_1 = xgb4_1.predict(X_test)
rmse_4_1 = np.sqrt(mean_squared_error(y_test, y_pred4_1))
r2_4_1 = r2_score(y_test, y_pred4_1)

rmse_4_1, r2_4_1
```

[ ]:
```
(1418.6095493847263, 0.9779383348861069)
```

As we can see, the negative RMSE decreased 3 points by setting lambda from 80 to 70. However, the mean absolute error for the test data predictions increased by 20 points. Thus, we decide to keep the lambda value at 70.

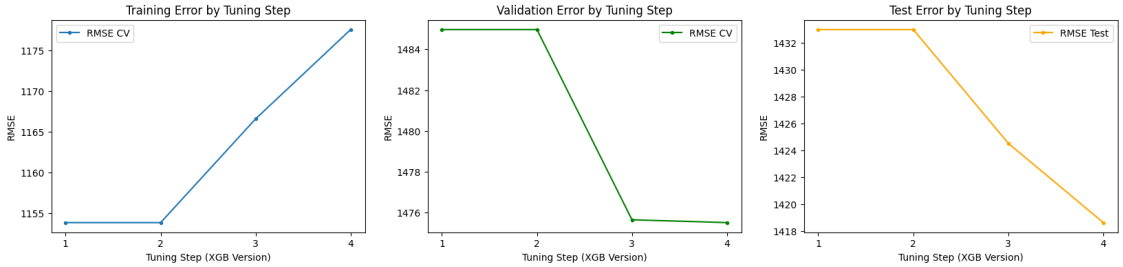
Evaluation of Hyperparameter Tuning

[ ]:
```
# Create lists with respective error values

# Error Metrics
neg_rmse_train = [np.min(xgb1_1.
 ↪cv_results_["mean_train_neg_root_mean_squared_error"]), np.min(xgb2.
 ↪cv_results_["mean_train_neg_root_mean_squared_error"]), np.min(xgb3.
 ↪cv_results_["mean_train_neg_root_mean_squared_error"]), np.min(xgb4_1.
 ↪cv_results_["mean_train_neg_root_mean_squared_error"])]
rmse_train = -1 * np.array(neg_rmse_train)
neg_rmse_cv = [xgb1_1.best_score_, xgb2.best_score_, xgb3.best_score_, xgb4_1.
 ↪best_score_]
rmse_cv = -1 * np.array(neg_rmse_cv)
rmse_test = [rmse_1_1, rmse_2, rmse_3, rmse_4_1]

# R2 Scores
r2_train = [np.min(xgb1_1.cv_results_["mean_train_r2"]), np.min(xgb2.
 ↪cv_results_["mean_train_r2"]), np.min(xgb3.cv_results_["mean_train_r2"]), np.
 ↪min(xgb4_1.cv_results_["mean_train_r2"])]
r2_cv = [np.min(xgb1_1.cv_results_["mean_test_r2"]), np.min(xgb2.
 ↪cv_results_["mean_test_r2"]), np.min(xgb3.cv_results_["mean_test_r2"]), np.
 ↪min(xgb4_1.cv_results_["mean_test_r2"])]
r2_test = [r2_1_1, r2_2, r2_3, r2_4_1]

# Custom X-Label according to tuning step
tuning_step = [1, 2, 3, 4]
```

[ ]:
```
# Create 1x3 figure grid for model performance in terms of error metric
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(20,4))
axes[0].set_title("Training Error by Tuning Step")
axes[0].plot(tuning_step, rmse_train, marker = ".", label = "RMSE CV")
axes[0].set_xticks(tuning_step)
axes[0].set_xlabel("Tuning Step (XGB Version)")
```

```
axes[0].set_ylabel("RMSE")
axes[0].legend()

axes[1].set_title("Validation Error by Tuning Step")
axes[1].plot(tuning_step, rmse_cv, marker = ".", label = "RMSE CV", color =␣
 ↪"green")
axes[1].set_xticks(tuning_step)
axes[1].set_xlabel("Tuning Step (XGB Version)")
axes[1].set_ylabel("RMSE")
axes[1].legend()

axes[2].set_title("Test Error by Tuning Step")
axes[2].plot(tuning_step, rmse_test, marker = ".", label = "RMSE Test", color =␣
 ↪"orange")
axes[2].set_xticks(tuning_step)
axes[2].set_xlabel("Tuning Step (XGB Version)")
axes[2].set_ylabel("RMSE")
axes[2].legend()
```

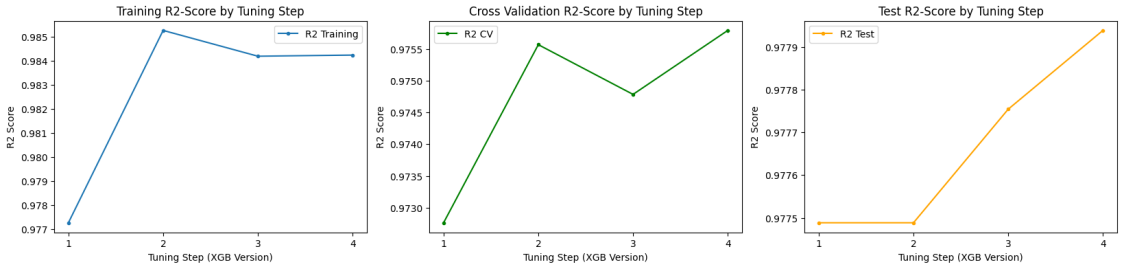
[ ]: <matplotlib.legend.Legend at 0x7c80636fb250>



[ ]:
```
# Create 1x3 figure grid for model performance in terms of error metric
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(20,4))
axes[0].set_title("Training R2-Score by Tuning Step")
axes[0].plot(tuning_step, r2_train, marker = ".", label = "R2 Training")
axes[0].set_xticks(tuning_step)
axes[0].set_xlabel("Tuning Step (XGB Version)")
axes[0].set_ylabel("R2 Score")
axes[0].legend()

axes[1].set_title("Cross Validation R2-Score by Tuning Step")
axes[1].plot(tuning_step, r2_cv, marker = ".", label = "R2 CV", color = "green")
axes[1].set_xticks(tuning_step)
axes[1].set_xlabel("Tuning Step (XGB Version)")
axes[1].set_ylabel("R2 Score")
axes[1].legend()
```

```
axes[2].set_title("Test R2-Score by Tuning Step")
axes[2].plot(tuning_step, r2_test, marker = ".", label = "R2 Test", color =␣
 ↪"orange")
axes[2].set_xticks(tuning_step)
axes[2].set_xlabel("Tuning Step (XGB Version)")
axes[2].set_ylabel("R2 Score")
axes[2].legend()
```

[ ]: <matplotlib.legend.Legend at 0x7a53c593ae60>



Both the respective error metrics and the R2 scroes indicate that every tuning step of the hyperparameters lead only to very little enhancement in model performance. However, the comparison between the validation error (RMSE) and the test error (MAE) as well as the evolution of the Test R2-Score support our decission of choosing the XGBoost Model with lambda set to 80. Even though the model performance increases for the validation sample, decreasing the L2 regularization parameter lambda to 70 yields a relatively strong decrease in model performance for the test sample.

Final XGBoost Model

[ ]:
```
# Define XGBoost with optimal parameters and fit on complete datset
xgb_agn= xgb.XGBRegressor(objective = "reg:squarederror", # Objective function␣
 ↪to be minimized by XGBoost model
                          enable_categorical = True,
                          eval_metric="rmse",
                          base_score = 0.5, # Initial prediction score for␣
 ↪all observations (default value).
                          booster = "gbtree", # Boosting type, here gradient␣
 ↪boosting tree
                          learning_rate = 0.1, # Learning rate of XGBoost␣
 ↪model determining the speed of convergence (default value)
                          n_estimators = 765, # Number of trees to be␣
 ↪estimated. Here, set to very high (unreached) number because stopped by␣
 ↪early_stopping_rounds.
```

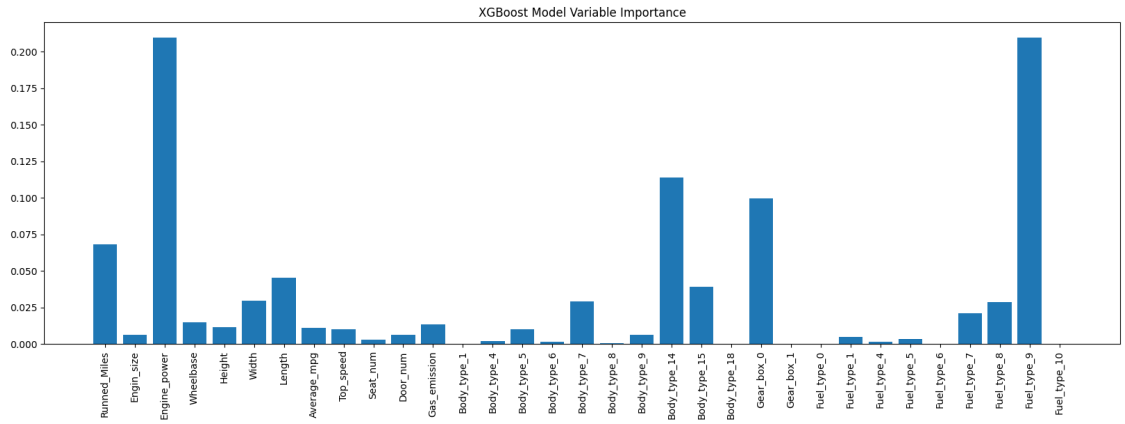
```
                            max_depth = 7, # Maximum tree depth (default value␣
↪= 6). Since we have p=14>6, we allow tree depth up to 8 levels. To be tuned␣
↪later.
                            min_child_weight = 1, # Minimum number of instances␣
↪needed per child (default value). To be tunded later.
                            reg_lambda = 1.5, # L2 regularization parameter␣
↪scaling the similiratiy scores and controlling for nodes containing only few␣
↪parameters. To be tuned later.
                            gamma = 0, # Similiarity score threshold (pruning␣
↪parameter) to create new split. To be tuned later.
                            subsample = 1, # Sample size of training data to be␣
↪used at each iteration. To be tuned, later.
                            sampling_method = "uniform", # Determines␣
↪distribution from which subsample is randomly drawn.
                            importance_type = "gain", # Criteria for the␣
↪model's feature_importance property
                            n_jobs = 7, # Number of CPU kernels to be used for␣
↪model training (time reduction)
                            verbosity = 2,
                            random_state = 999
                            )

xgb_agn.fit(X_train,y_train)
```

[ ]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=True, eval_metric='rmse', feature_types=None,
             gamma=0, grow_policy=None, importance_type='gain',
             interaction_constraints=None, learning_rate=0.1, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=7, max_leaves=None,
             min_child_weight=1, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=765, n_jobs=7,
             num_parallel_tree=None, random_state=999, …)
```

C) Model Agnostic Methods Basic Variable Importance First we plot the feature importance based on the default importance criterion which is gain. More precisely, to obtian the feature importance the gain in simliartiy provided by creating a new node (binary split) based on a specific feature is calculated (averaged).

[ ]:
```
# Create Bar chart displaying the variable importance for our XGBoost Model
names = X.columns
values_imp = xgb_agn.feature_importances_

plt.figure(figsize=(20,6))
```

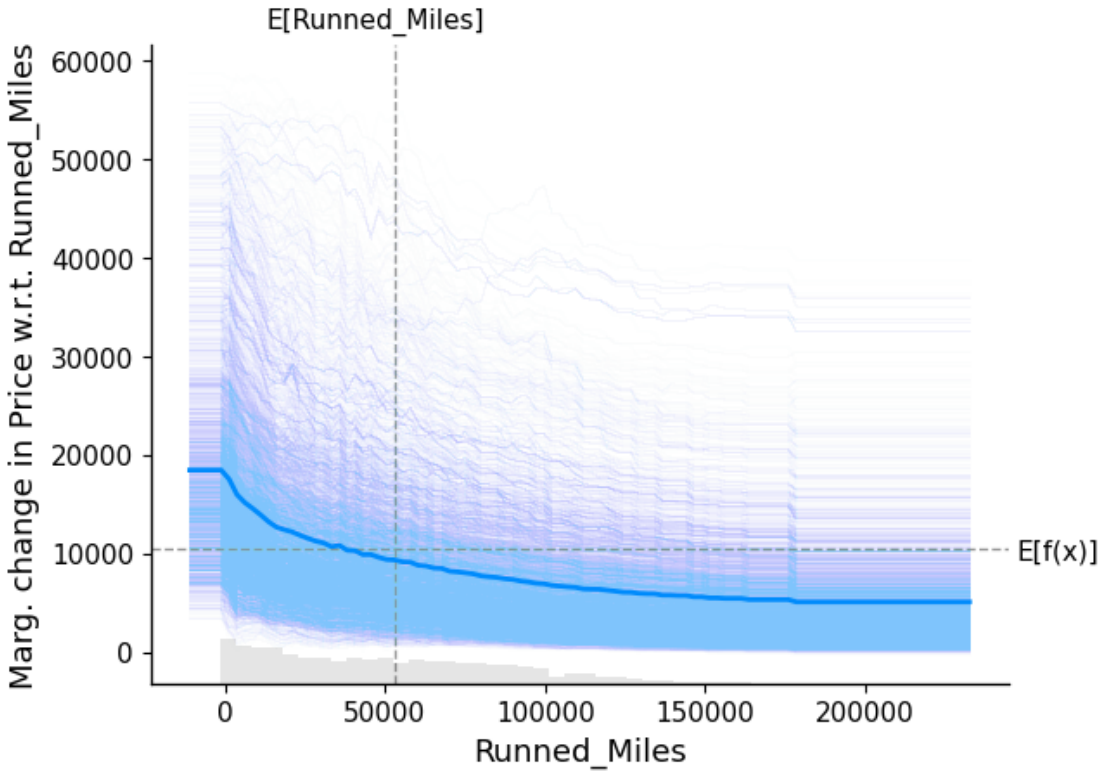
```
plt.bar(names, values_imp)
plt.xticks(rotation=90)
plt.title("XGBoost Model Variable Importance")
plt.show()
```

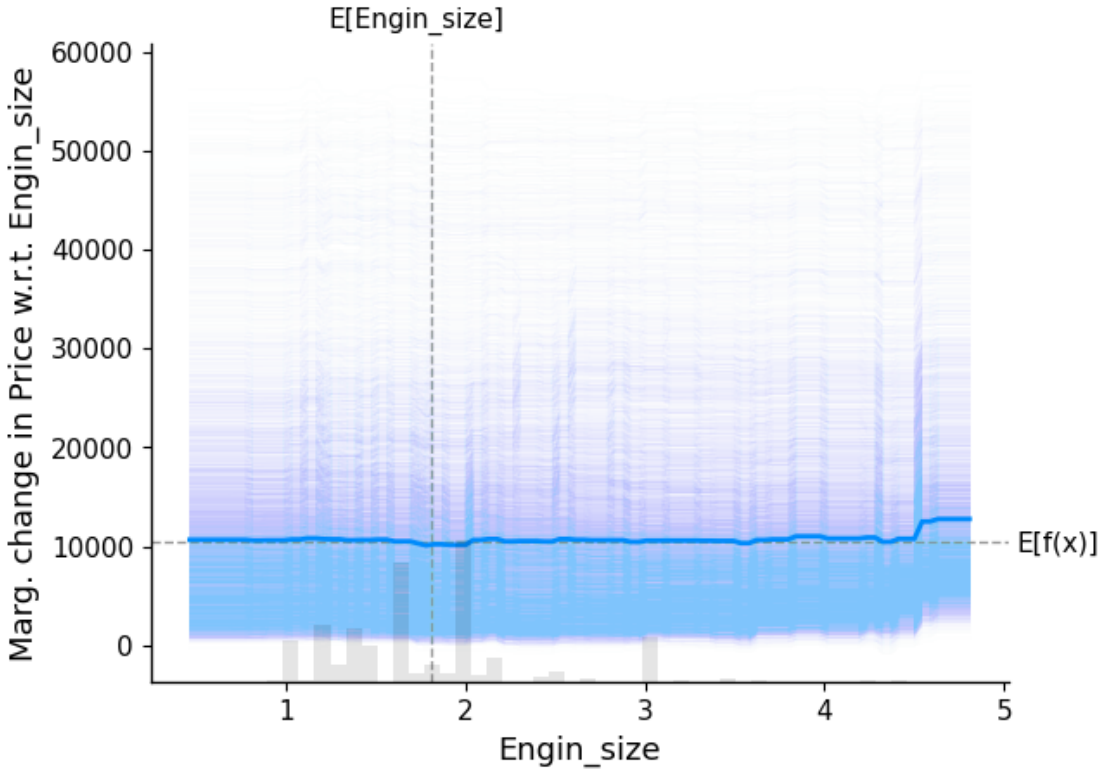


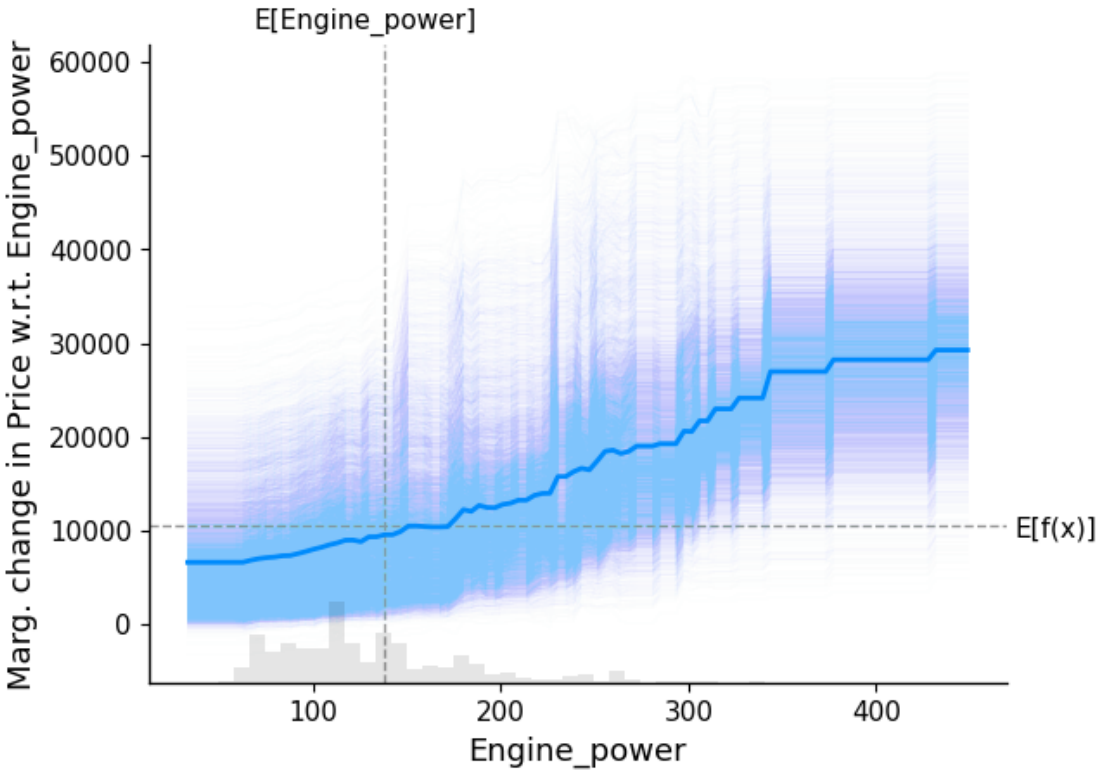
Partial Dependence Plots

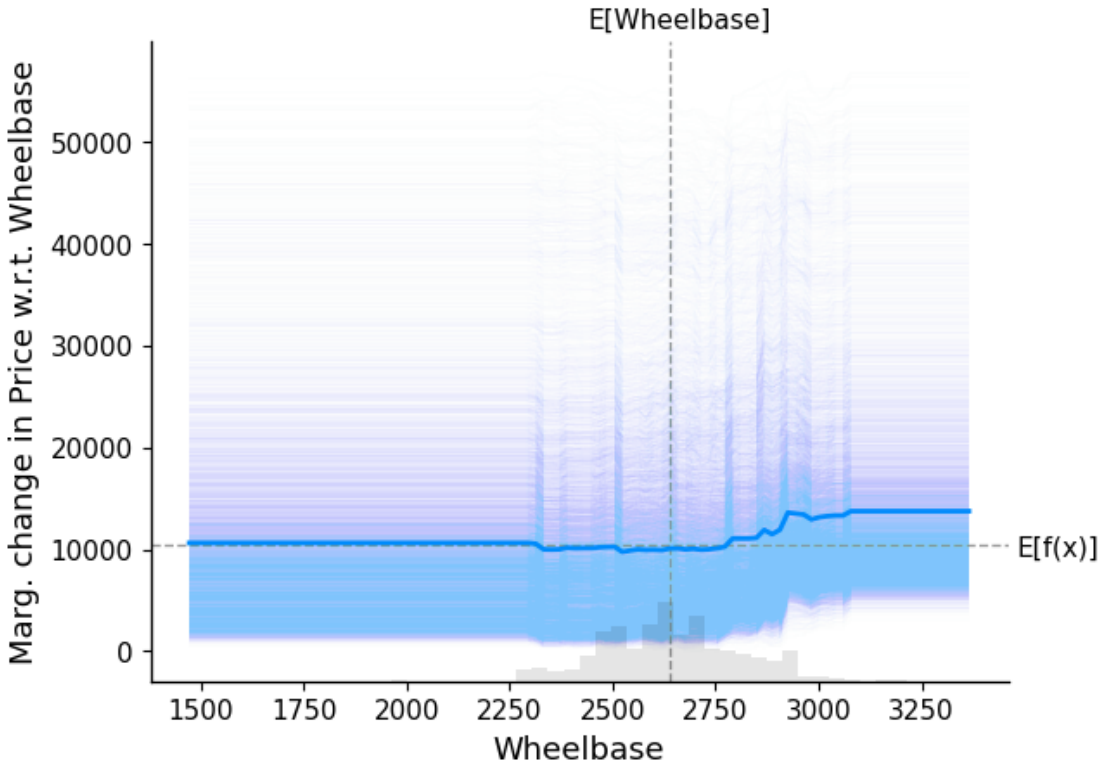
[ ]:
```
# Create background distirbution as reference
X_ref = shap.utils.sample(X, 50000)

# Define features for which we want to plot a pdp
features = X.columns
```

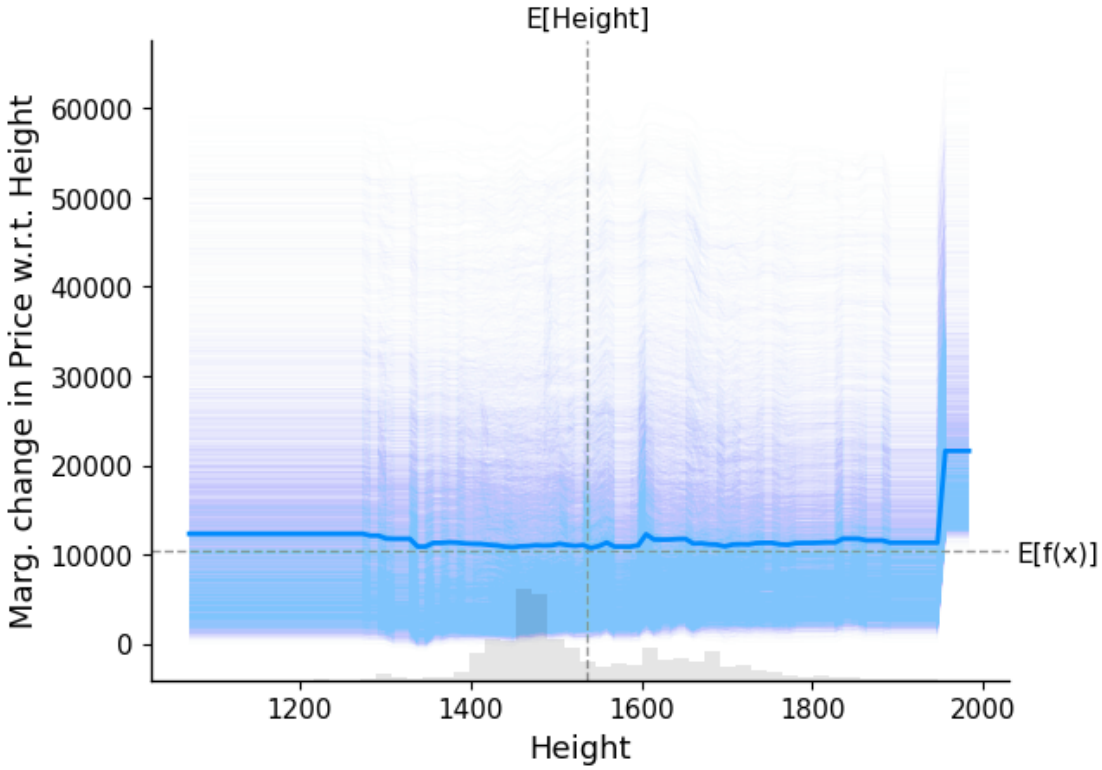
[ ]:
```
# Start loop and create PDPs for desired features
for feature in features:
  shap.partial_dependence_plot(
      ind = feature,
      model = xgb_agn.predict,
      data = X_ref,
      model_expected_value = True,
      feature_expected_value = True,
      ice = True,
      show = True,
      ylabel = f'Marg. change in Price w.r.t. {feature}'
  )
```

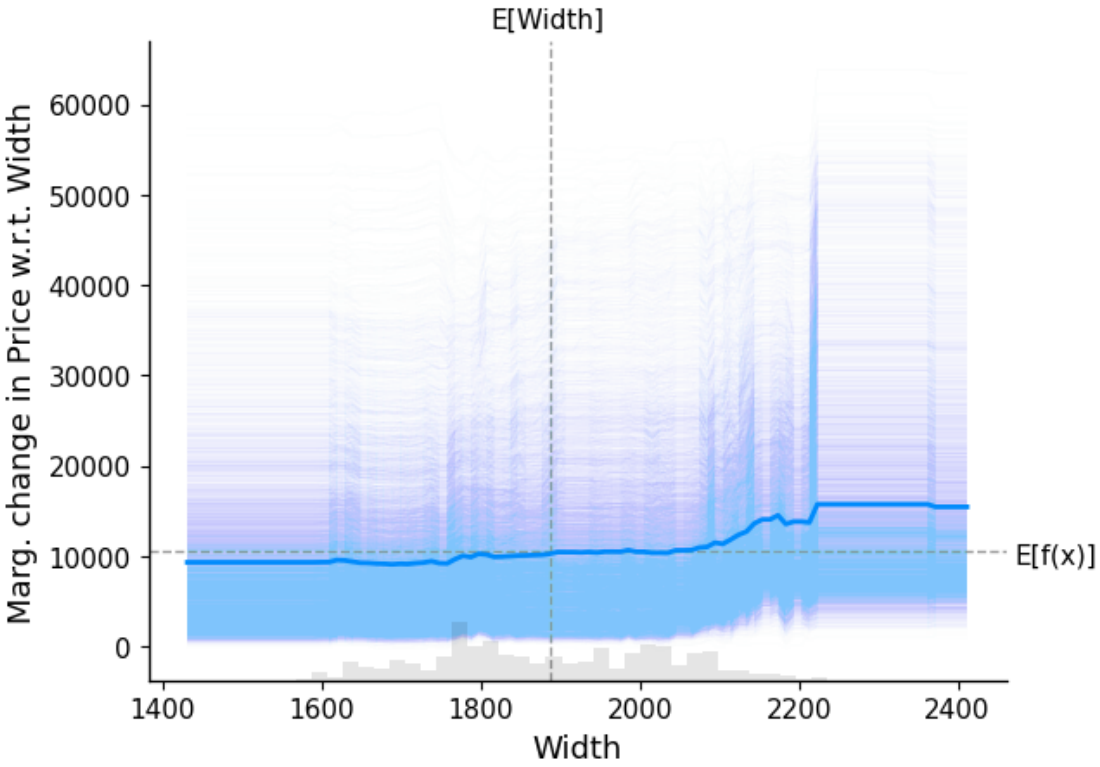

E[Runned_Miles]
Marg. change in Price w.r.t. Runned_Miles
60000
50000
40000
30000
20000
10000
0
E[f(x)]
0
50000
100000
150000
200000
Runned_Miles

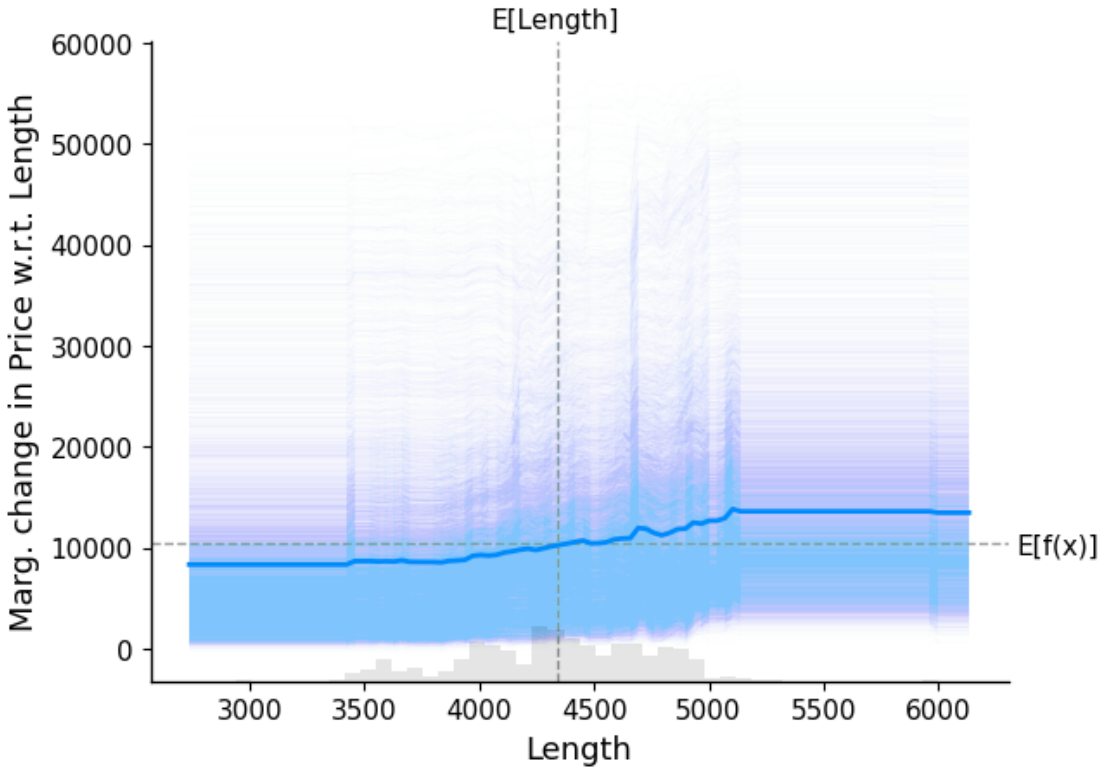

E[Engin_size]
Marg. change in Price w.r.t. Engin_size
60000
50000
40000
30000
20000
10000
0
E[f(x)]
1
2
3
4
5
Engin_size

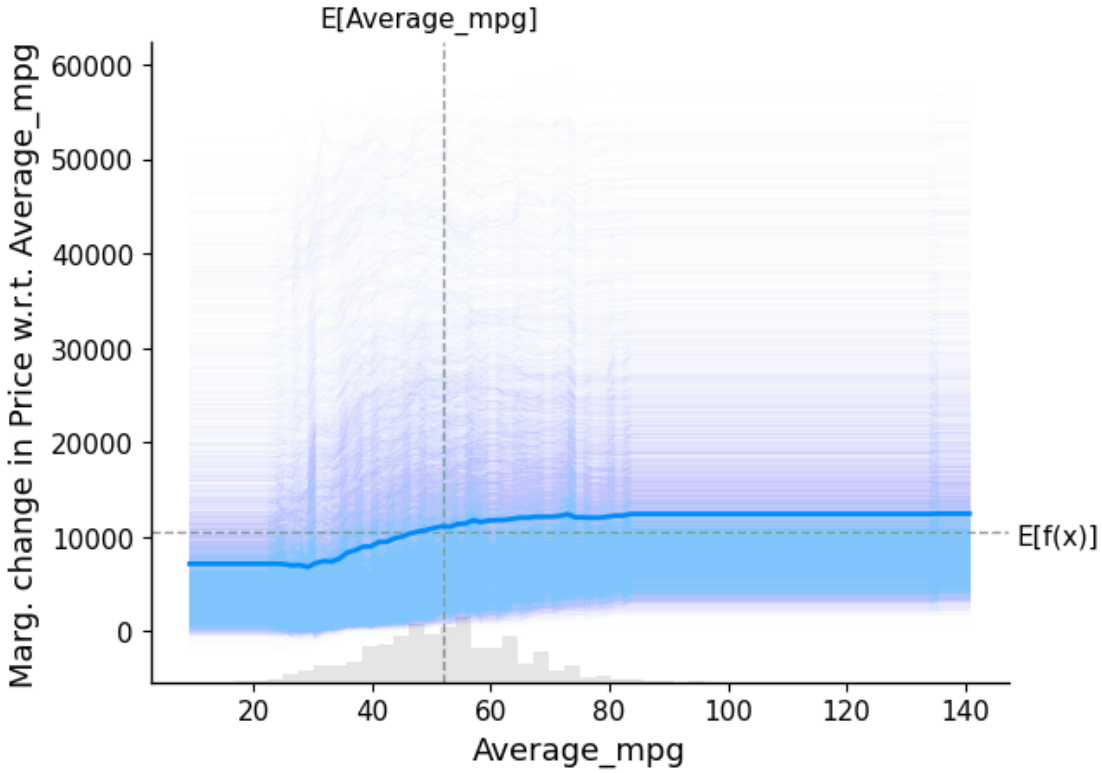

E[Engine_power]
Marg. change in Price w.r.t. Engine_power
60000
50000
40000
30000
20000
10000
0
E[f(x)]
100
200
300
400
Engine_power

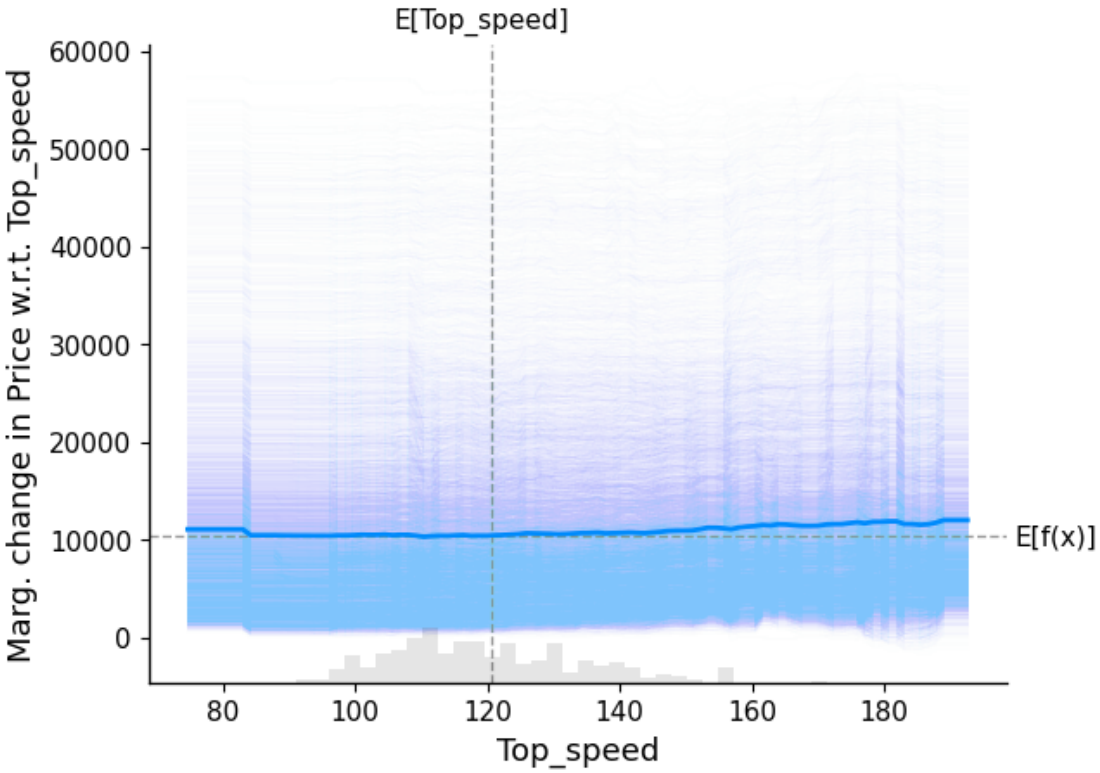

E[Wheelbase]
Marg. change in Price w.r.t. Wheelbase
50000
40000
30000
20000
10000
0
E[f(x)]
1500
1750
2000
2250
2500
2750
3000
3250
Wheelbase

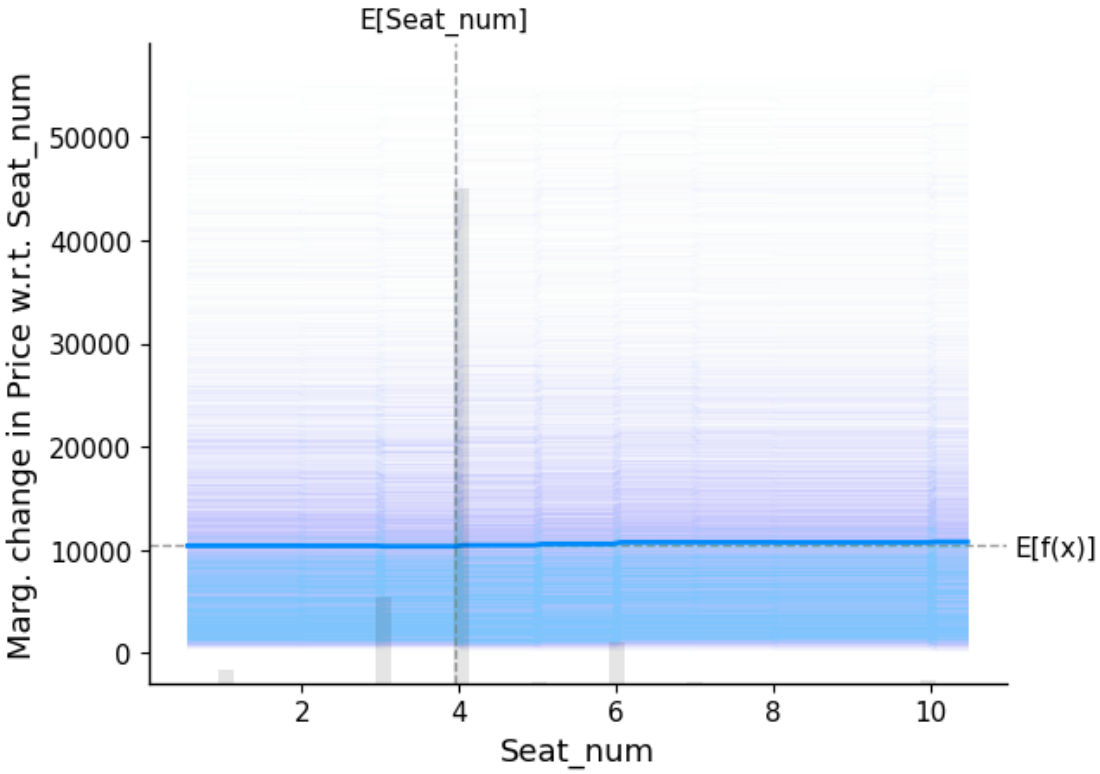

E[Height]
Marg. change in Price w.r.t. Height
60000
50000
40000
30000
20000
10000
0
E[f(x)]
1200
1400
1600
1800
2000
Height

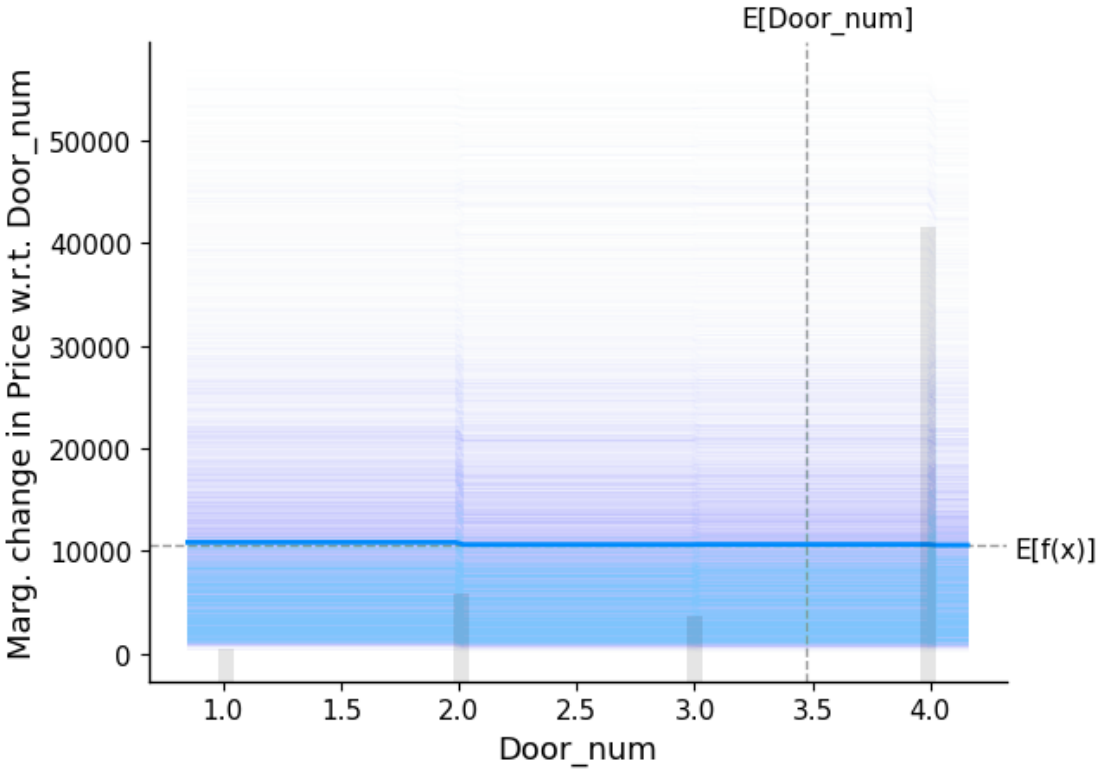

E[Width]
Marg. change in Price w.r.t. Width
60000
50000
40000
30000
20000
10000
0
E[f(x)]
1400
1600
1800
2000
2200
2400
Width

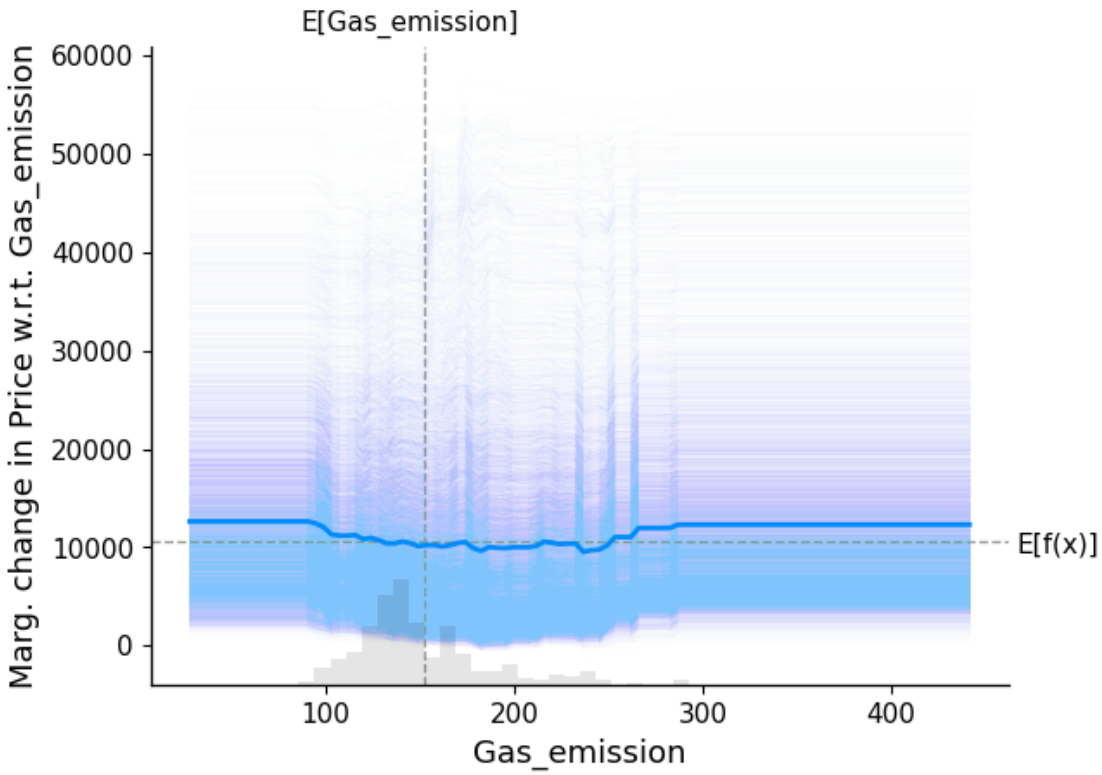

E[Length]
60000
50000
40000
30000
20000
10000
0
Marg. change in Price w.r.t. Length
E[f(x)]
3000
3500
4000
4500
5000
5500
6000
Length


E[Average_mpg]
60000
50000
40000
30000
20000
10000
0
Marg. change in Price w.r.t. Average_mpg
E[f(x)]
20
40
60
80
100
120
140
Average_mpg

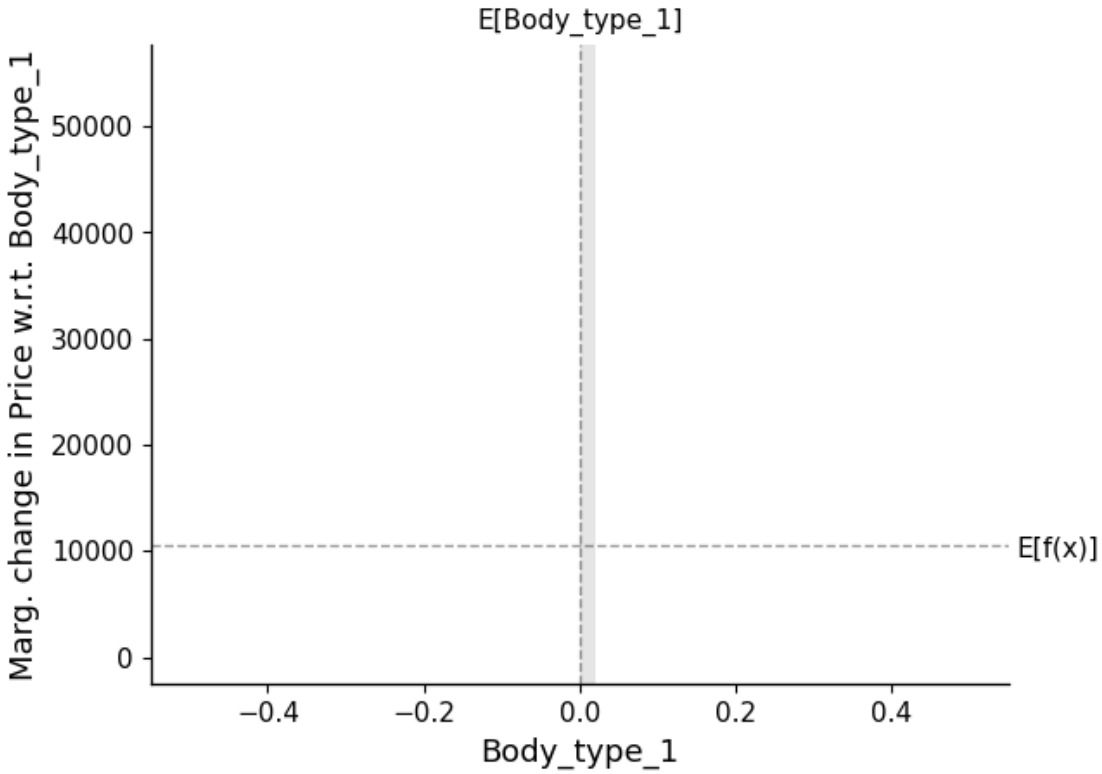

E[Top_speed]
60000
50000
40000
30000
20000
10000
0
Marg. change in Price w.r.t. Top_speed
E[f(x)]
80
100
120
140
160
180
Top_speed

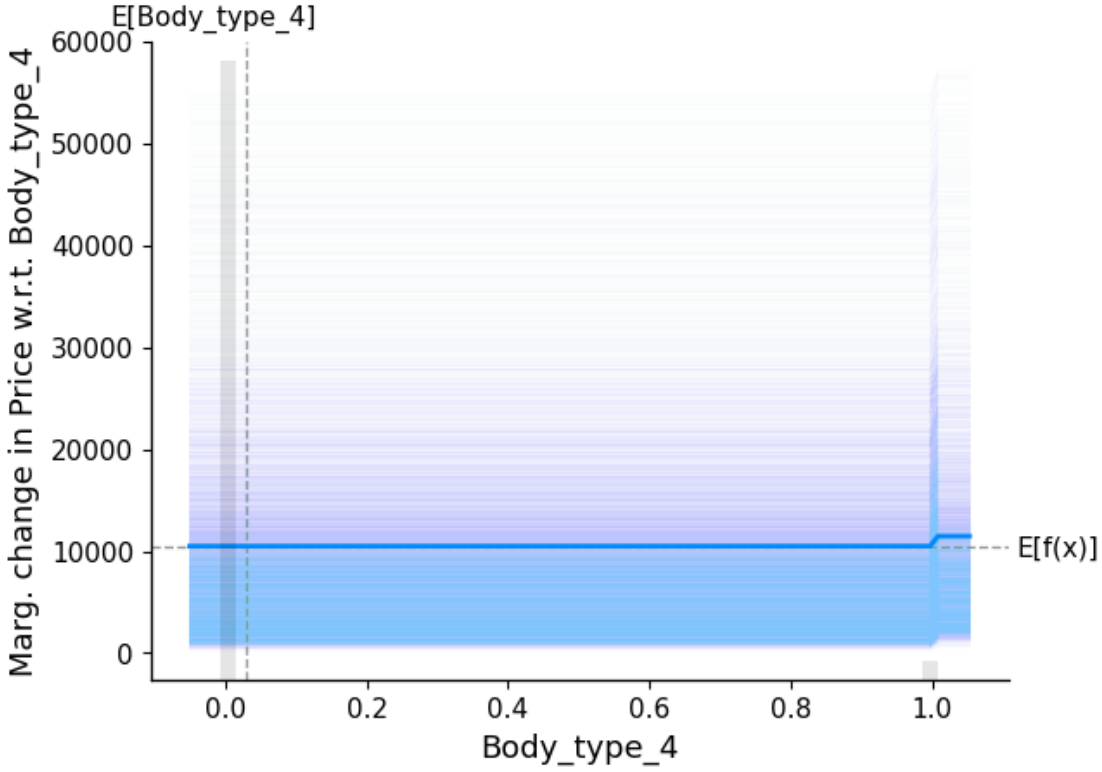

E[Seat_num]
Marg. change in Price w.r.t. Seat_num
50000
40000
30000
20000
10000
0
E[f(x)]
2
4
6
8
10
Seat_num

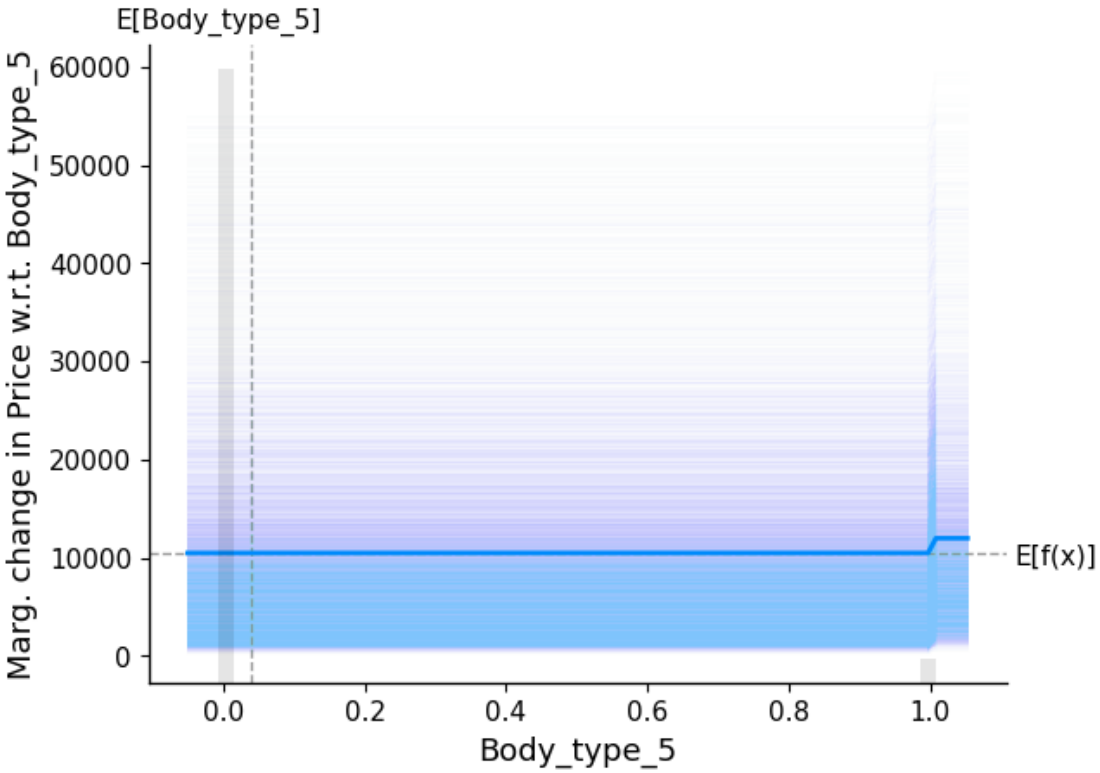

E[Door_num]
Marg. change in Price w.r.t. Door_num
50000
40000
30000
20000
10000
0
E[f(x)]
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Door_num

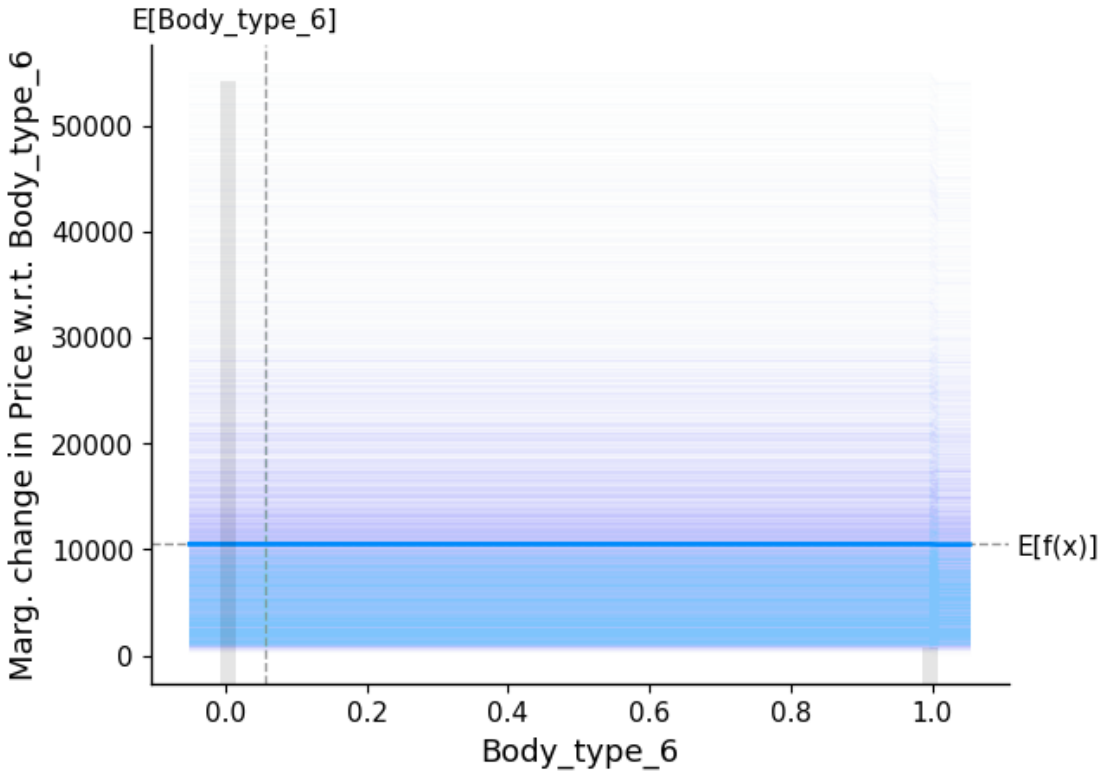


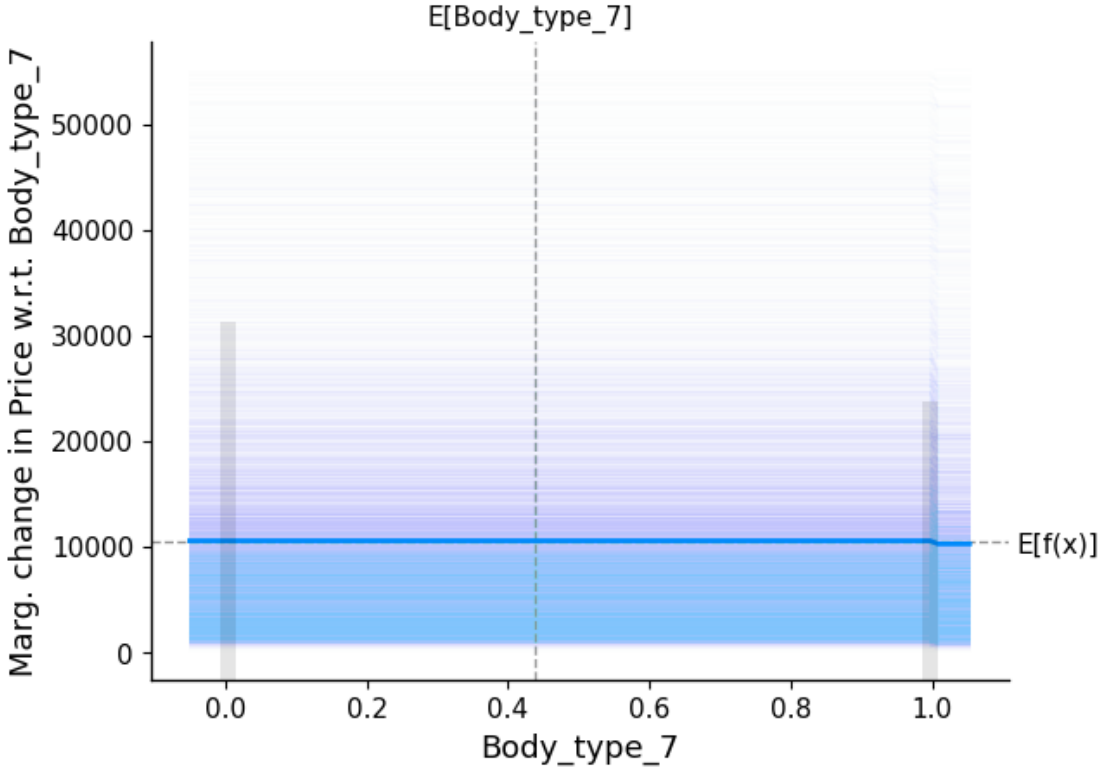

```
Attempting to set identical low and high xlims makes transformation singular;
automatically expanding.
```

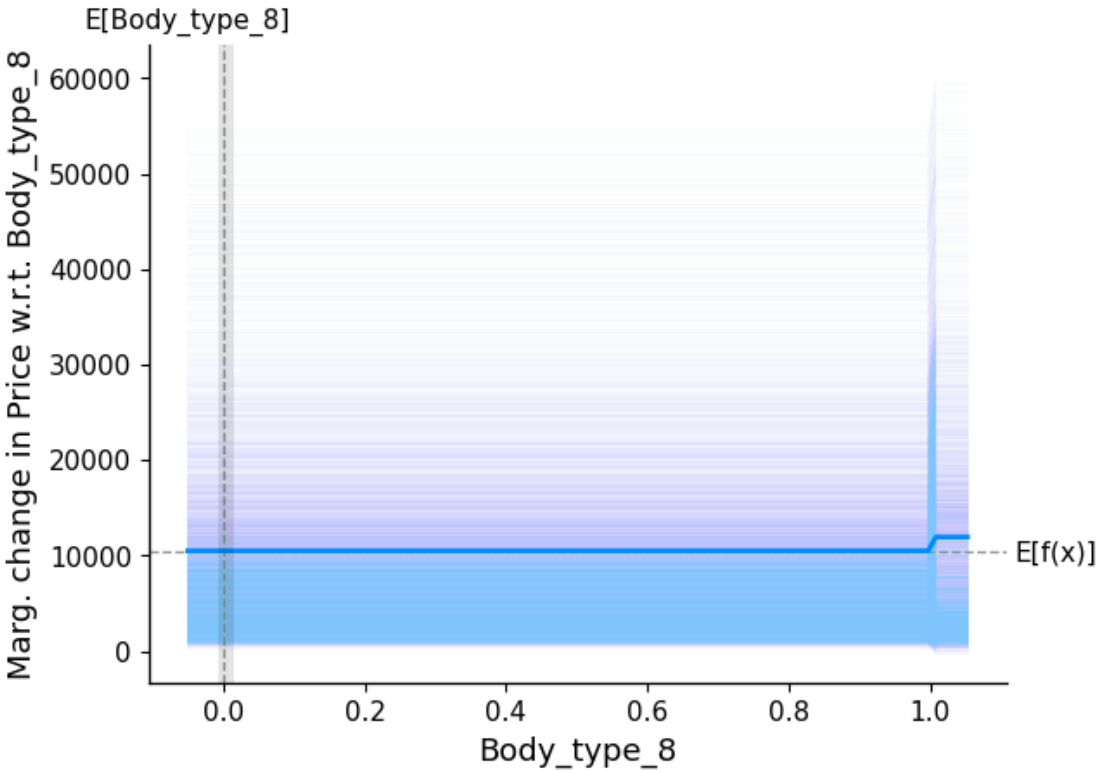

E[Body_type_1]
Marg. change in Price w.r.t. Body_type_1
50000
40000
30000
20000
10000
0
E[f(x)]
−0.4
−0.2
0.0
0.2
0.4
Body_type_1

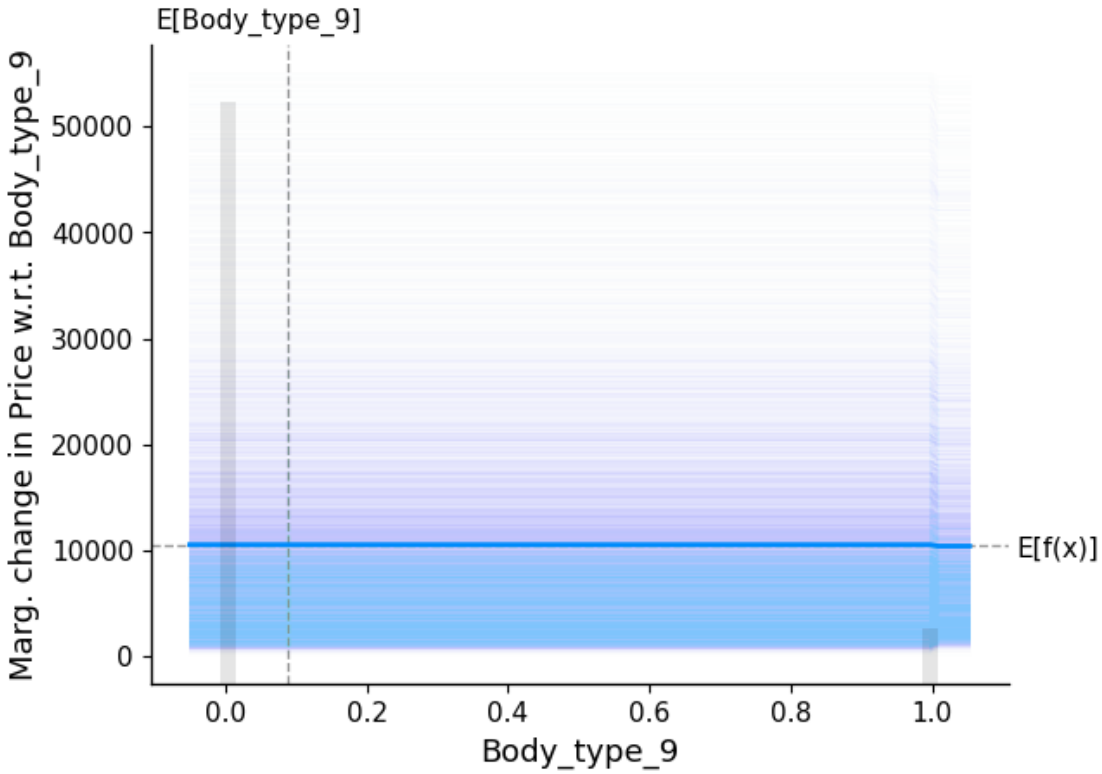

E[Body_type_4]
Marg. change in Price w.r.t. Body_type_4
60000
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_4

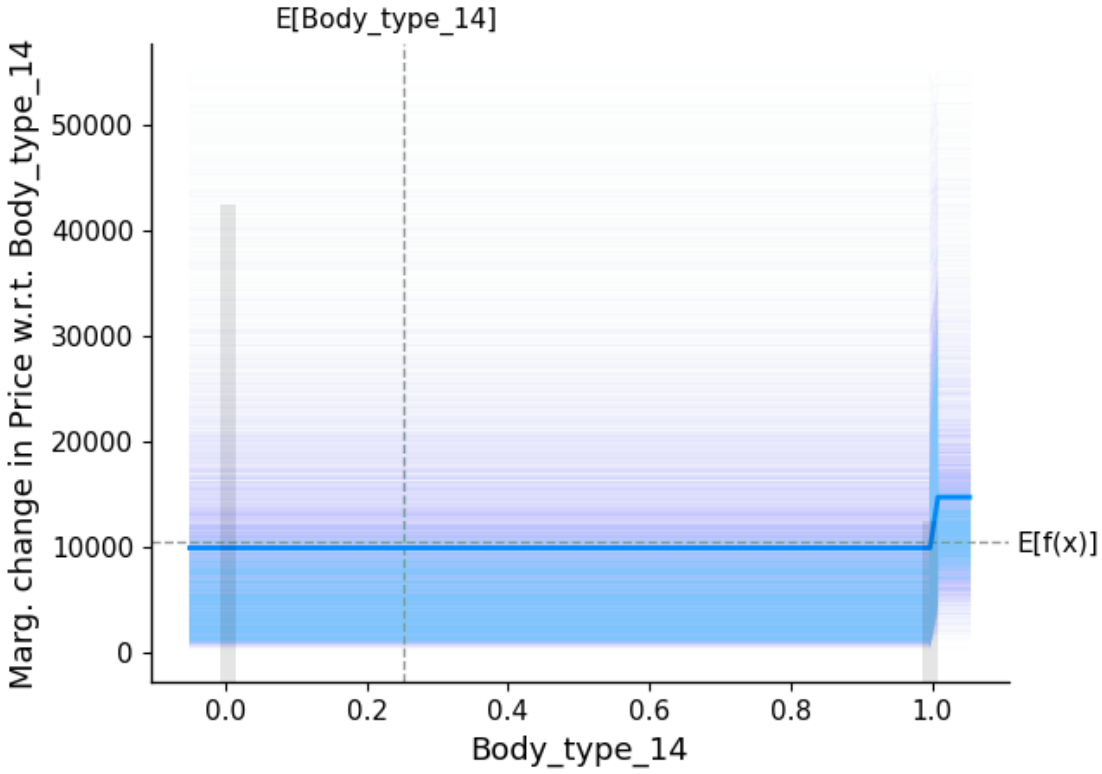

E[Body_type_5]
Marg. change in Price w.r.t. Body_type_5
60000
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_5

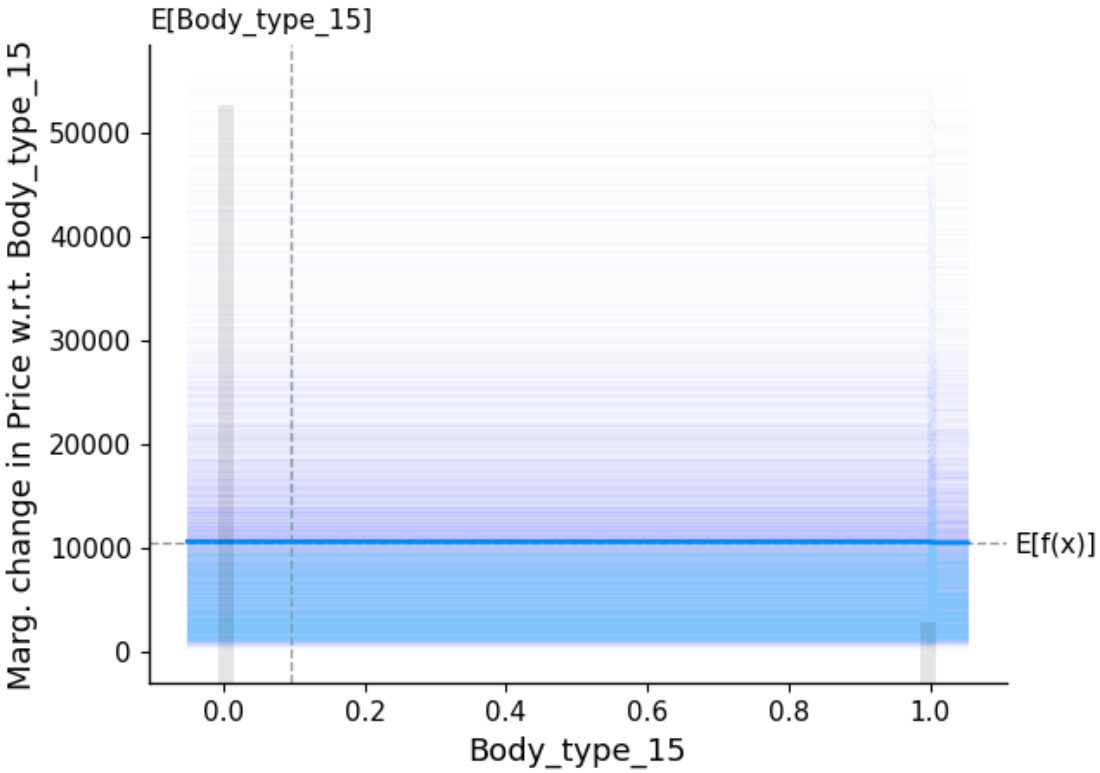

E[Body_type_6]
Marg. change in Price w.r.t. Body_type_6
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_6

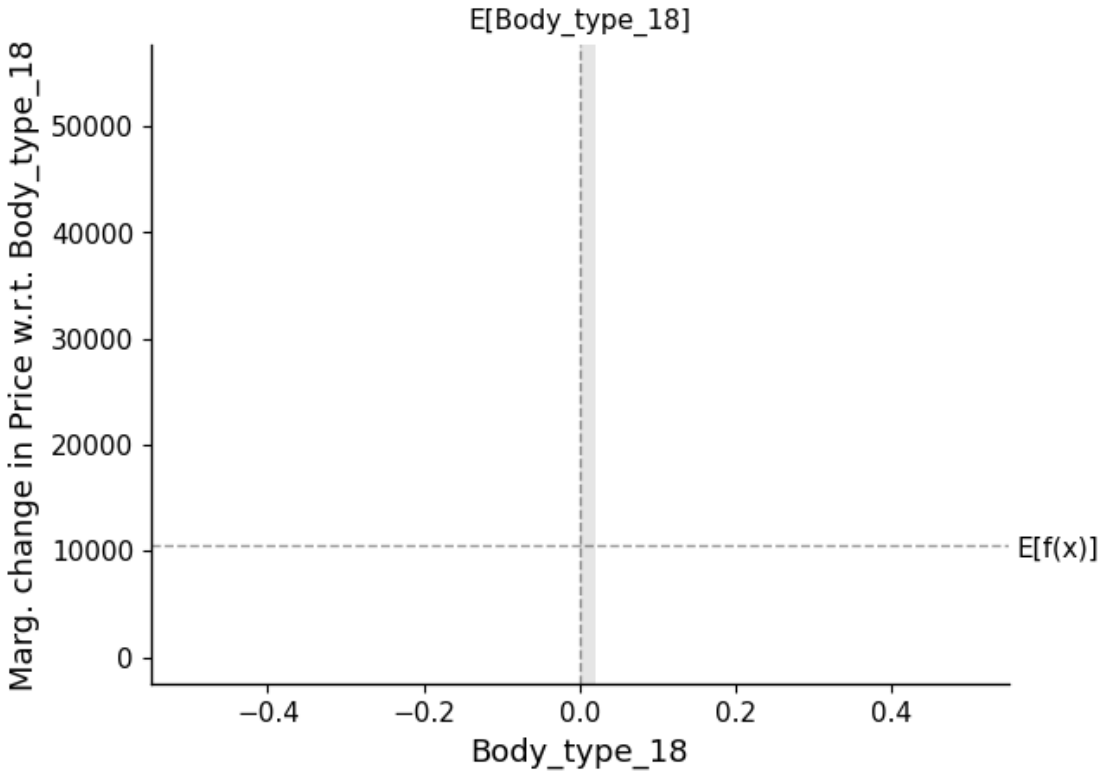

E[Body_type_7]
Marg. change in Price w.r.t. Body_type_7
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_7

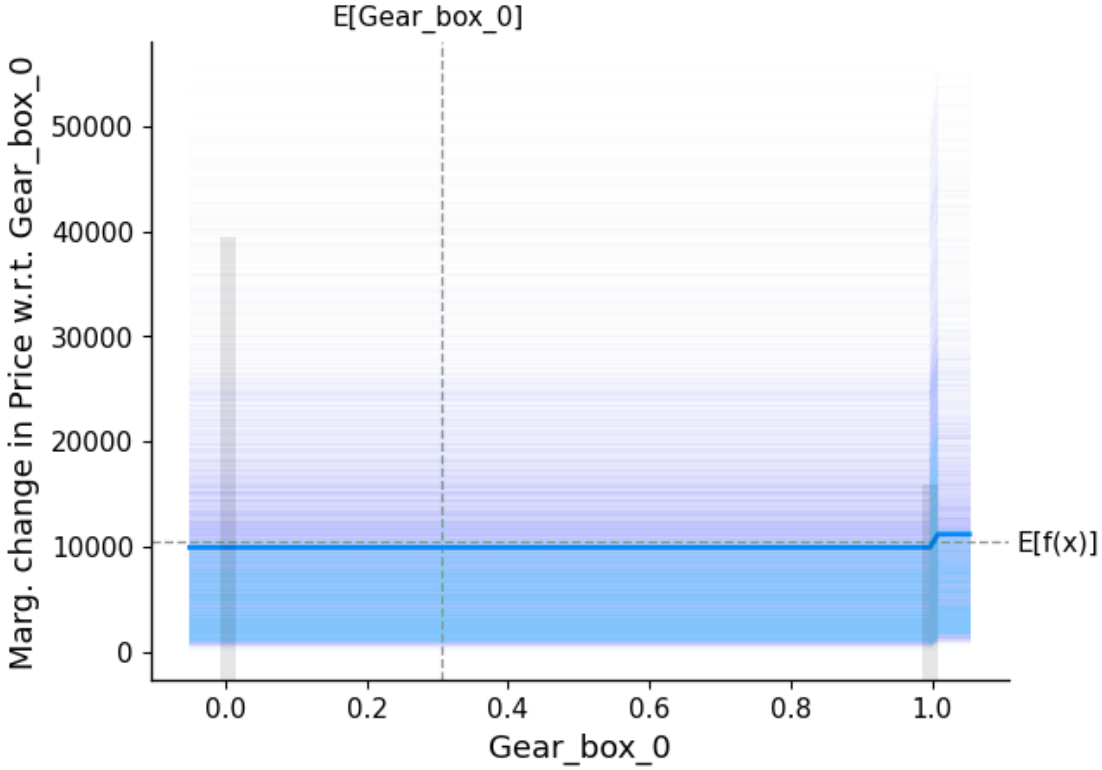

E[Body_type_8]
Marg. change in Price w.r.t. Body_type_8
60000
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_8

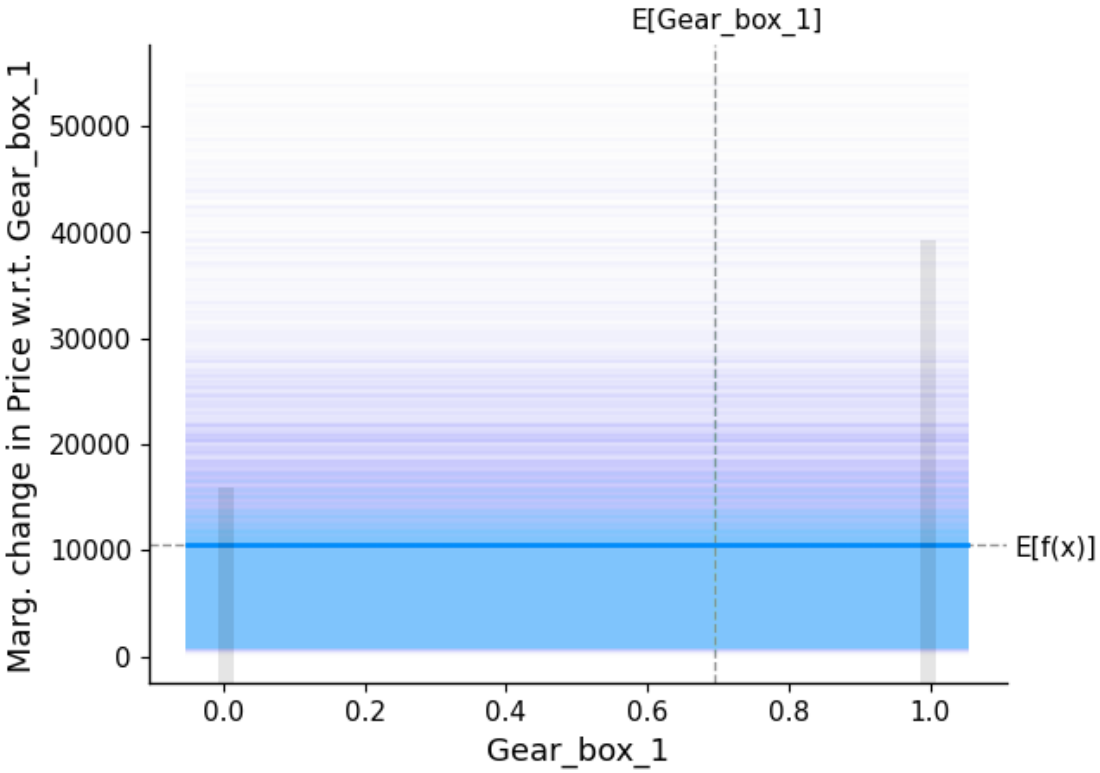

E[Body_type_9]
Marg. change in Price w.r.t. Body_type_9
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_9

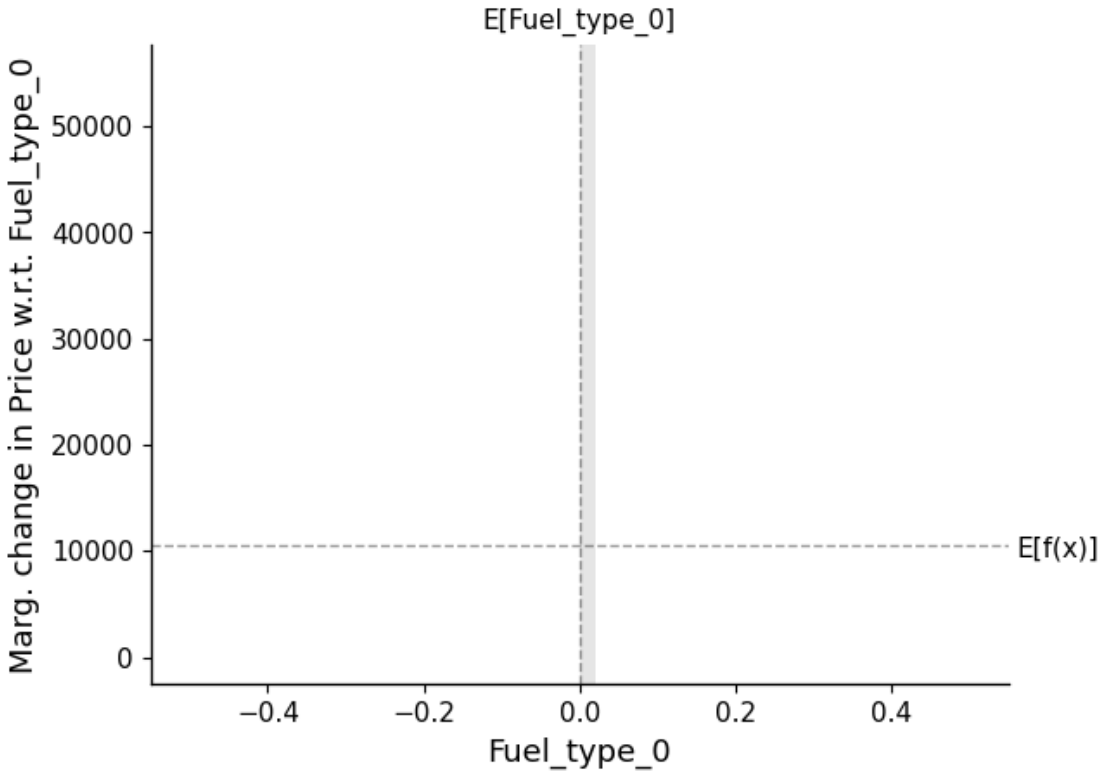

E[Body_type_14]
Marg. change in Price w.r.t. Body_type_14
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_14

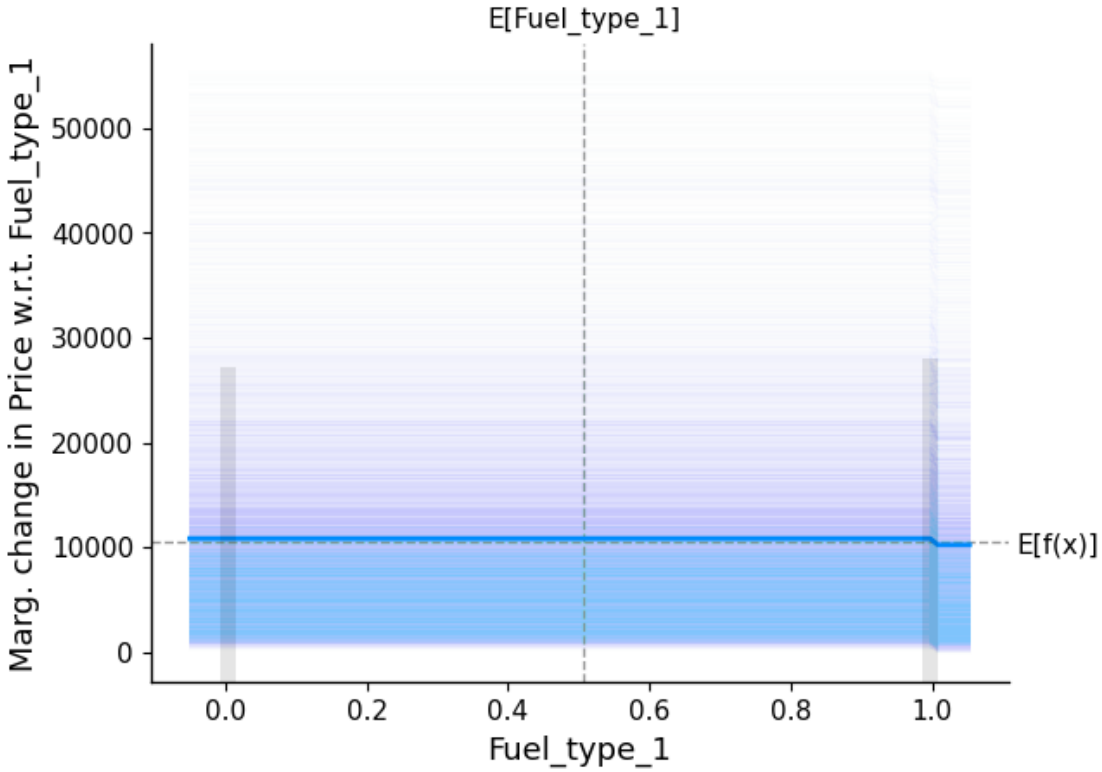

E[Body_type_15]
Marg. change in Price w.r.t. Body_type_15
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_15

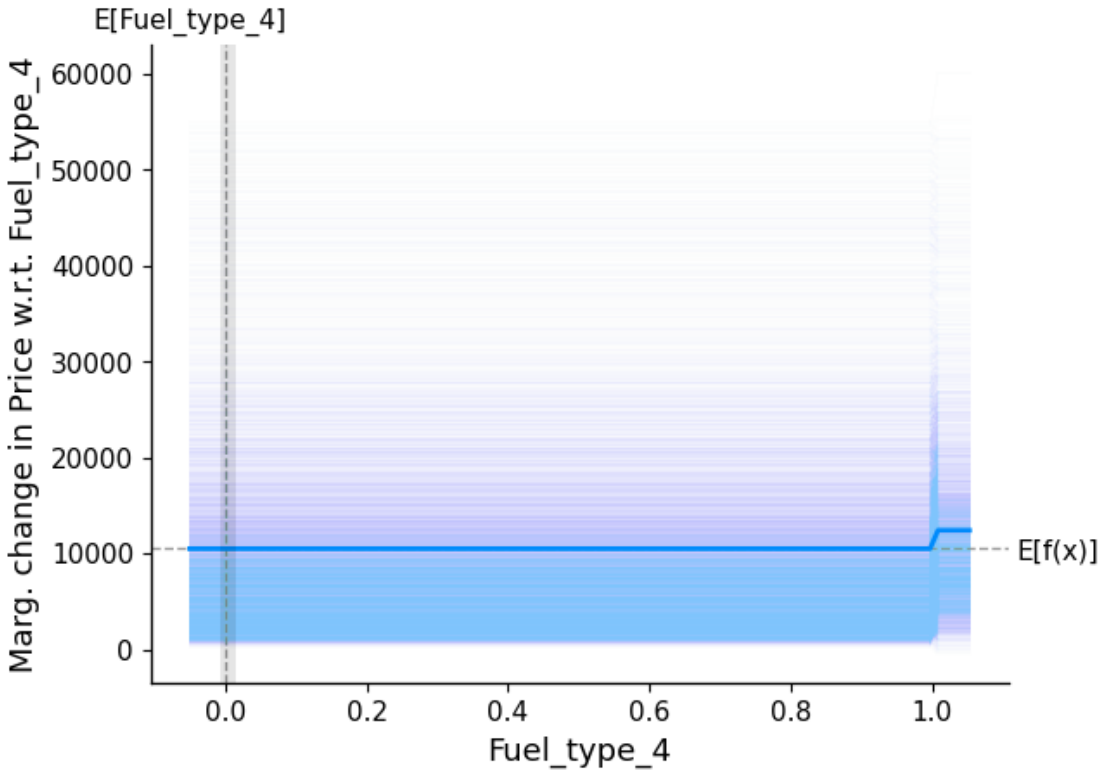

E[Body_type_18]
Marg. change in Price w.r.t. Body_type_18
50000
40000
30000
20000
10000
0
E[f(x)]
−0.4
−0.2
0.0
0.2
0.4
Body_type_18

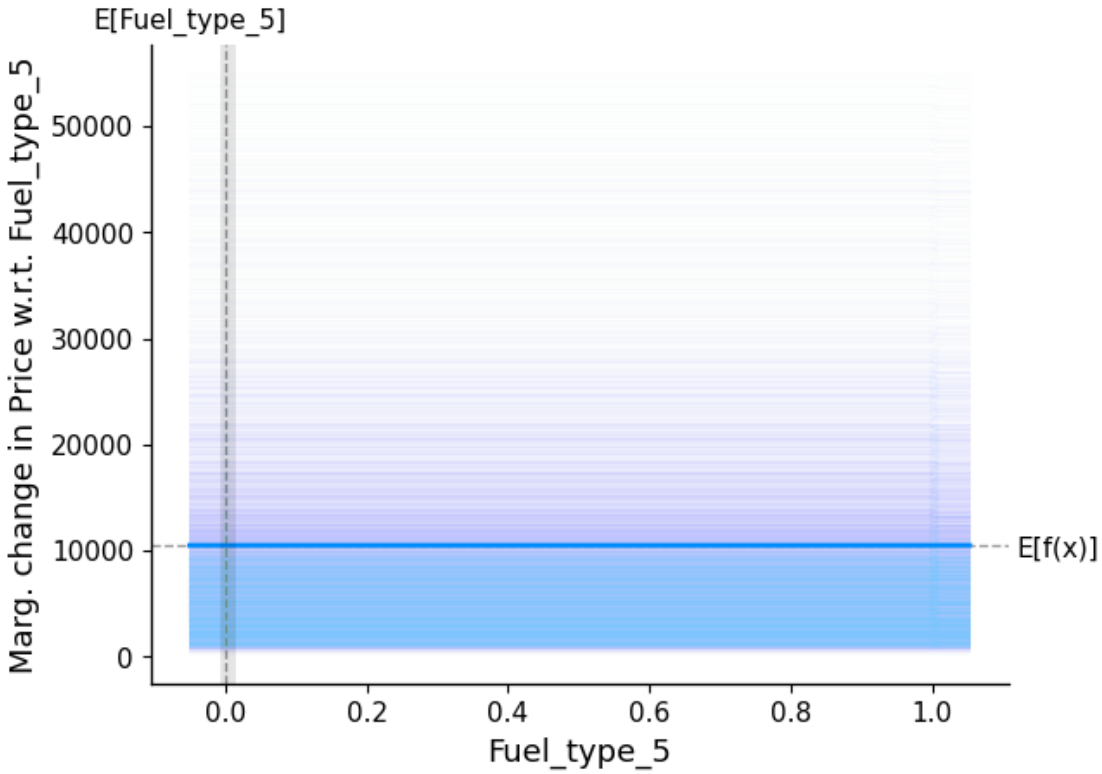

E[Gear_box_0]
Marg. change in Price w.r.t. Gear_box_0
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Gear_box_0

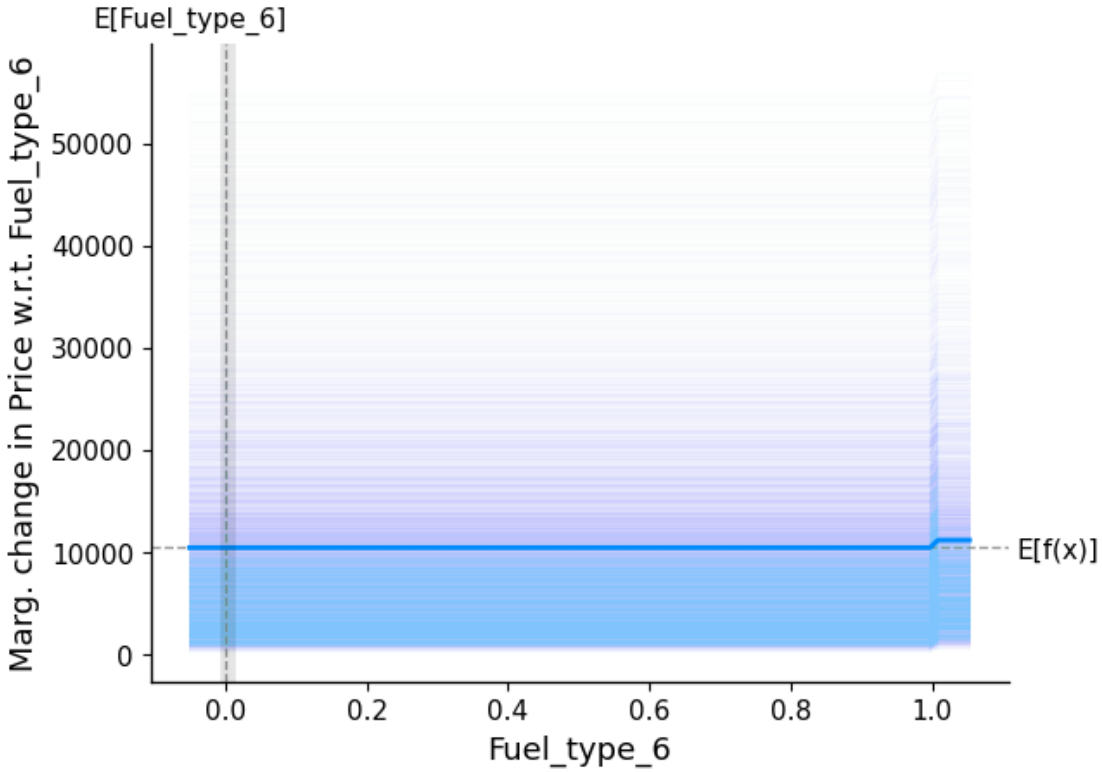

E[Gear_box_1]
Marg. change in Price w.r.t. Gear_box_1
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Gear_box_1

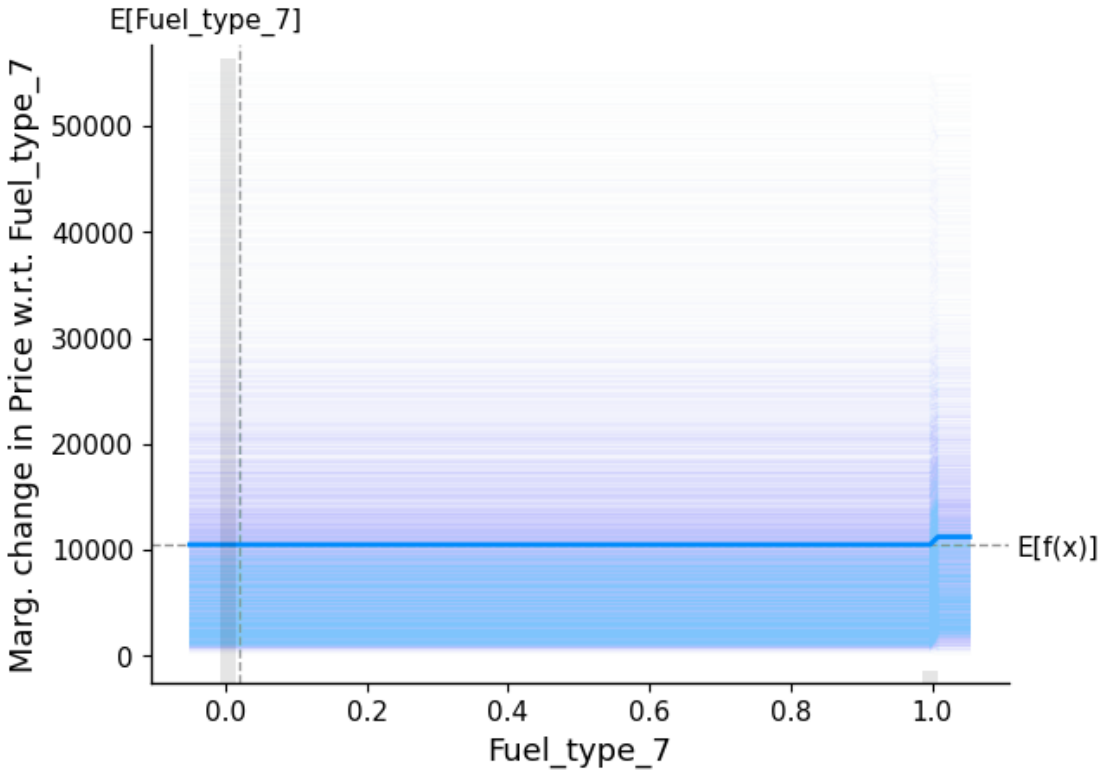

E[Fuel_type_0]
Marg. change in Price w.r.t. Fuel_type_0
50000
40000
30000
20000
10000
0
E[f(x)]
−0.4
−0.2
0.0
0.2
0.4
Fuel_type_0

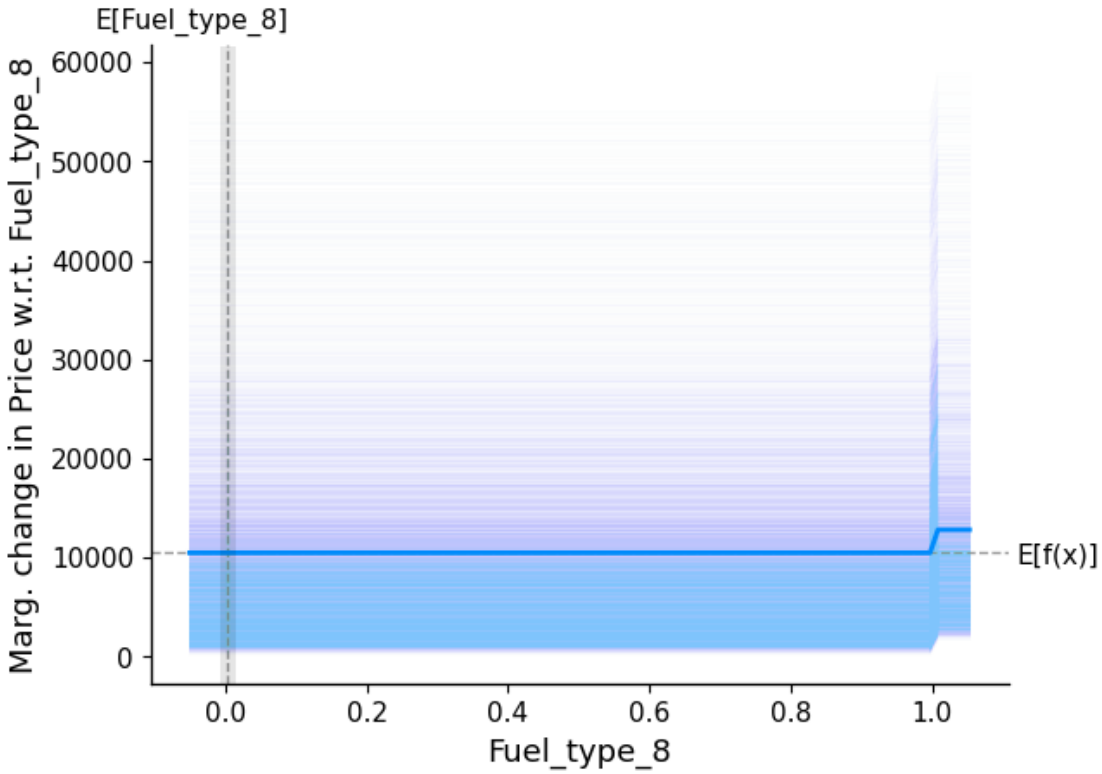

E[Fuel_type_1]
Marg. change in Price w.r.t. Fuel_type_1
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Fuel_type_1

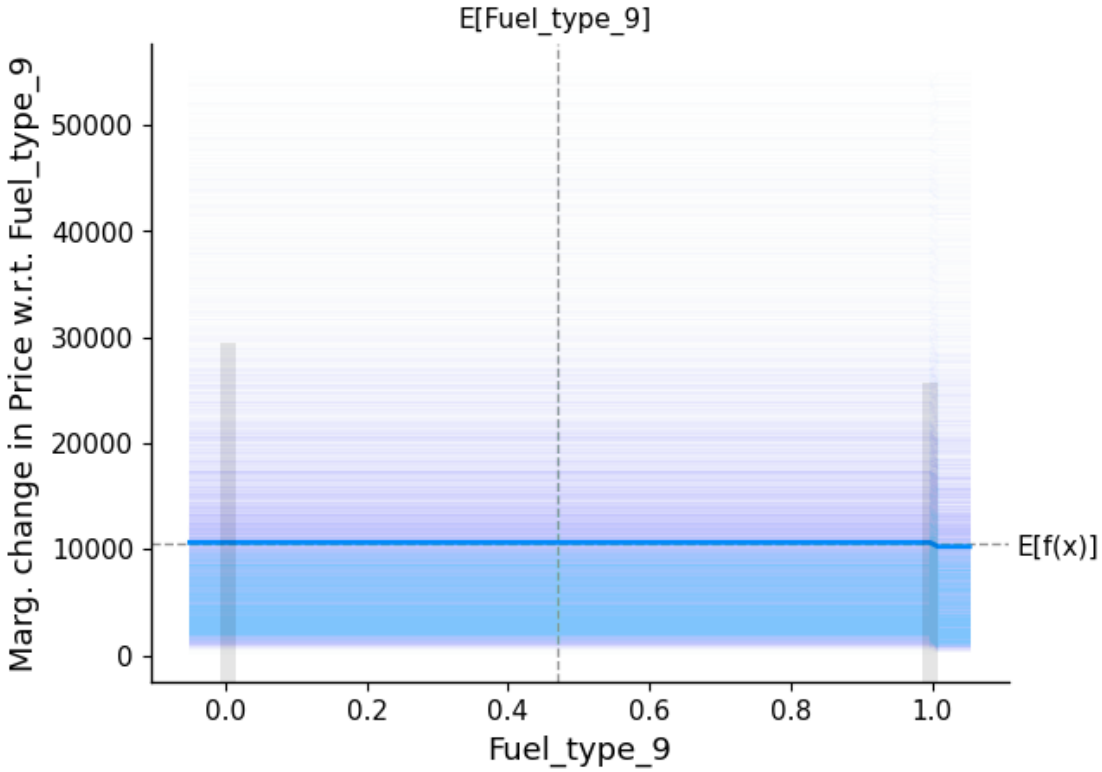

E[Fuel_type_4]
Marg. change in Price w.r.t. Fuel_type_4
60000
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Fuel_type_4

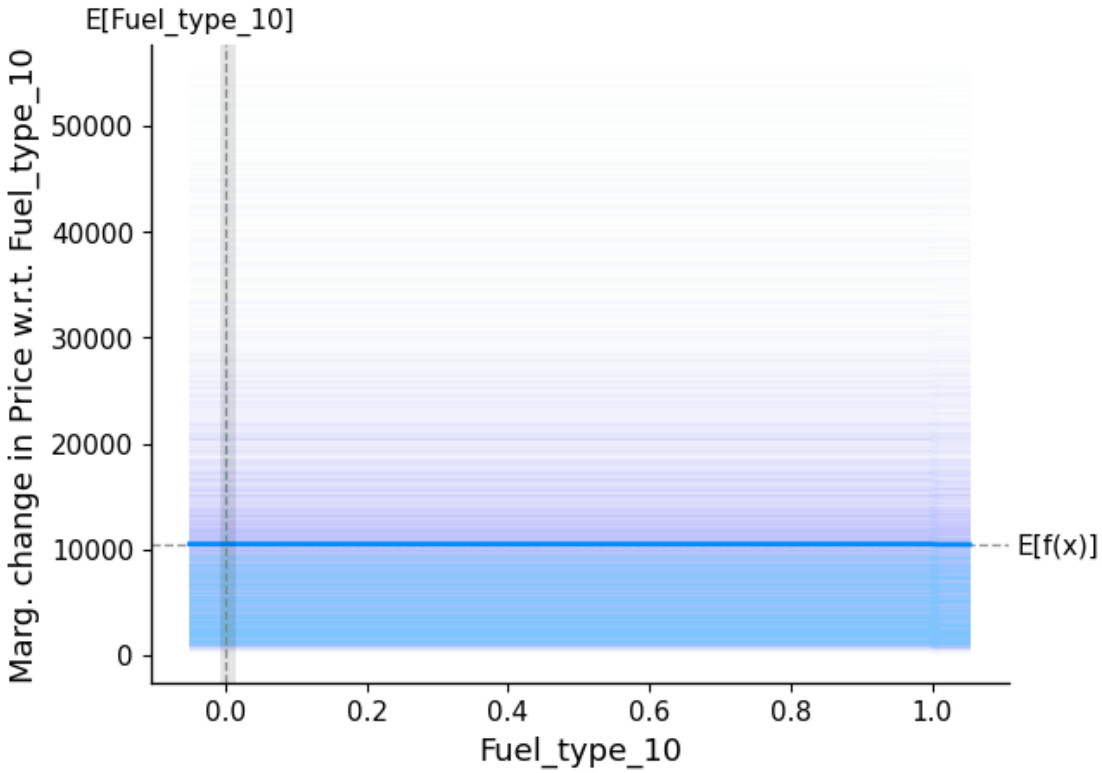

E[Fuel_type_5]
Marg. change in Price w.r.t. Fuel_type_5
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Fuel_type_5


E[Fuel_type_6]
Marg. change in Price w.r.t. Fuel_type_6
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Fuel_type_6


E[Fuel_type_7]
Marg. change in Price w.r.t. Fuel_type_7
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Fuel_type_7


E[Fuel_type_8]
Marg. change in Price w.r.t. Fuel_type_8
60000
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Fuel_type_8


E[Fuel_type_9]
Marg. change in Price w.r.t. Fuel_type_9
50000
40000
30000
20000
10000
0
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
Fuel_type_9



For every feature, the ICE curves follow the same curves, indicating no significant intersction effects between the variables for predicting the car price. Thus, the PDP plor might already be a sufficient globale model-agnostic tool to capture the relationship between our features and the target variable.

Shapely Values

[ ]:
```
# Import necessary library to display shap plots
from IPython.display import display, Javascript
display(Javascript('initjs()'))

# import javascript visualization library
shap.initjs()
```

```
<IPython.core.display.Javascript object>
```

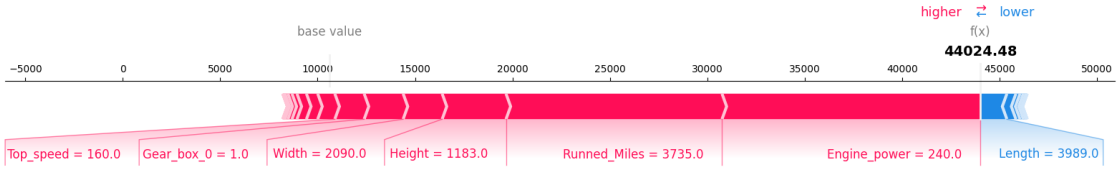
```
<IPython.core.display.HTML object>
```

[ ]:
```
# Define explainer object for the final XGBoost Model
explainer = shap.TreeExplainer(xgb_agn)

# Compute Shap values for all features using the small subsample X_ref
```

```
shap_values = explainer.shap_values(X_ref)
```

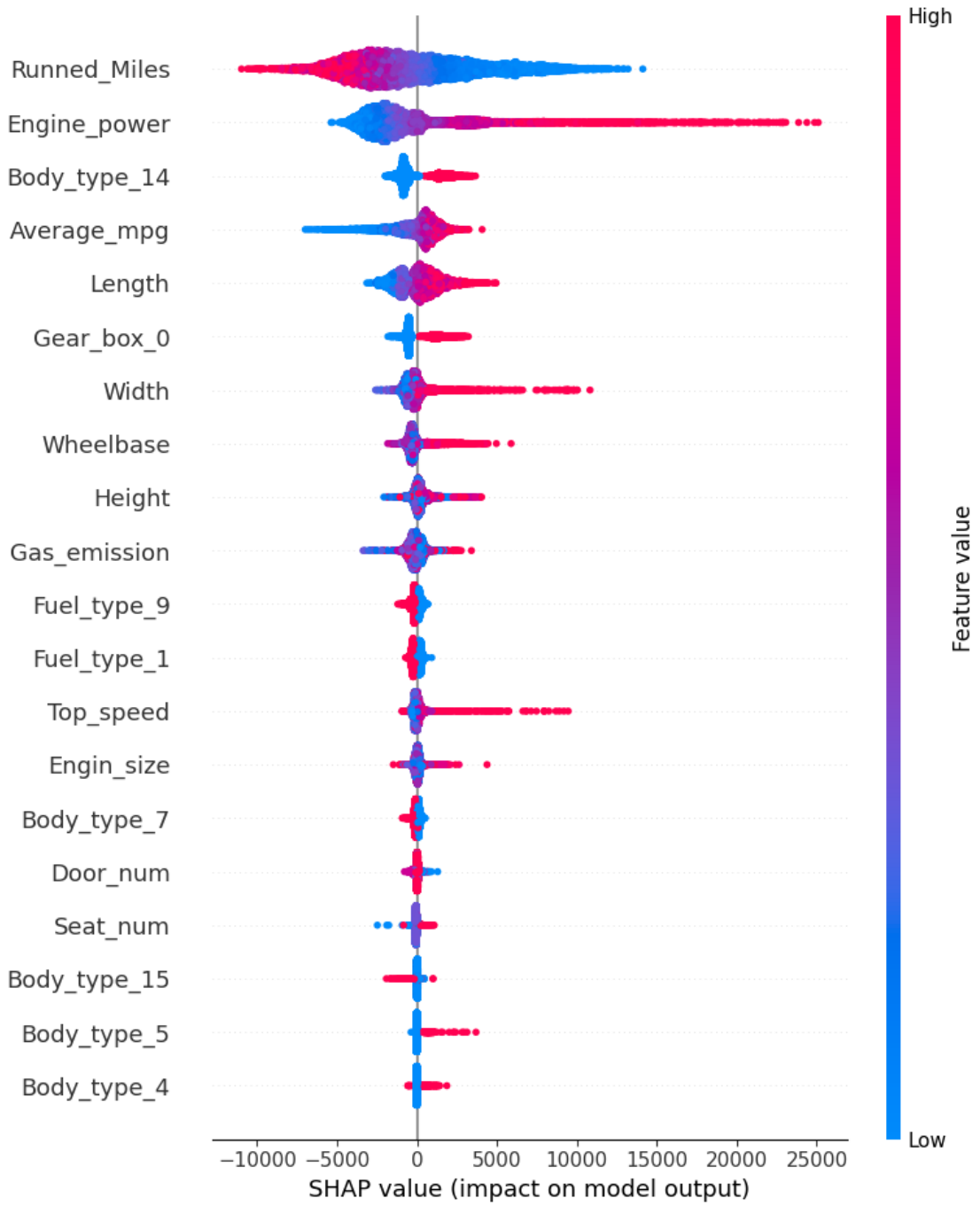
```
[18:44:36] WARNING: /workspace/src/c_api/c_api.cc:1240: Saving into deprecated
binary model format, please consider using `json` or `ubj`. Model format will
default to JSON in XGBoost 2.2 if not specified.
```

[ ]:
```
# Create shap force plot to visualize the first occurance prediction's
 ↪explanation
shap.force_plot(explainer.expected_value,
                shap_values[0,:],
                X_ref.iloc[0,:],
                matplotlib=True
                #contribution_threshold = # controls features displayed on the
 ↪plot.
                )
```



This force graph explains the prediction of our XGBoost model for the first observation in our data. We can observe that for this car, the low milage (13,479) and the high enigine power (335 hp) have a positive influence on the car's price evaluation. In contrast, the relatively low top speed of 155 mp/h and the car's height (high, low, inch, cm, intepretation?) reduce the car's price. While the base value represents the expected value (i.e. average prediction over all input instances), f(x) denotes the observation's predicted price, driven by the red and blue force arrows.

[ ]:
```
# Use Beeswarm plot to summarize the shap values (impact on model output) for
 ↪all features
shap.summary_plot(shap_values, X_ref)
```



As the summary beeswarm graph shows, we can see that especially engine power, runned miles, height, top speed as well as engine size drive our model results. More precisely, the red and blue colors indicate that our model predicts a higher price for cars with much horse power, few runned miles, a low overall height and high top speed. The plot also displays that cars with many runned miles will be evaluated at a lower price, i.e. high number of runned miles negatively impacts a car's price prediction.

Comparing the Shapely plots with the standard variable importance function from the scikit-learn library, we can see that both approaches determine engine power, top speed, runned miles and height (in descending order) as most significant features for the car price estimation.

D) Information Reduction Method: Elastic Net XGBoost by construction prunes and accounts for its complexity, i.e. controls for overfitting. However, in the following we want to challenge and apply an information reduction method, namely the elastic net. By fitting an elastic net, we can determine the most important variables and refit our XGBoost model using this subselection of features. Again, since the XGBoost algorithm automatically creates its binary splits based on those features providing the most gain in smilarity score, this approach is not expected to improve the model performance but may even slightly worsen our results. The later might happen because we might exclude variables that have low predictive power for a linear model but might be a feature for a chosen candiadate split for the XGBoost. That however is but very rather unlikely.

Since we are in a high dimension case with 33 features, we can apply so called information reduction methods that will reduce the number of features considered in our models. From our correlation matrix, we can observe that features which are logically related to each other like specifictions on the engine or car dimensions show high correlation coefficients. Thus, we can use an elastic net as reduction method which is a combination of the ridge and lasso to obtain the most important features for predicting the car prices. However, before applying the elastic net, we need to standardize the data such that the l1 and l2 penalization are equally applied to all our variables, i.e. all features are on the same scale

[ ]:
```
from sklearn.preprocessing import StandardScaler
# Standardize the training data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[ ]:
```
# Define the hyperparameter grid
param_grid_elastic = {
    'max_iter': [1000],
    'alpha': np.arange(0.0, 1.0, 0.1),
    'l1_ratio': np.arange(0.0, 1.0, 0.1)
}

# Create an ElasticNet model
elastic_reg = ElasticNet(random_state=999)

# Create a GridSearchCV object
elastic = GridSearchCV(estimator = elastic_reg,
                       param_grid = param_grid_elastic,
                       scoring = ["neg_root_mean_squared_error", "r2"],
                       refit = "neg_root_mean_squared_error",
                       n_jobs = 7, # Define number of CPU cores to use on your␣
 ↪machine
                       cv = 4,
                       verbose = 3,
```

```
                    return_train_score = True, # Set to true such that the␣
 ↪cv_results will also include the training error scores
                    )

# Fit the model to the training data
elastic.fit(X_train_scaled, y_train)

# Access the best hyperparameters and the best model
best_params = elastic.best_params_
best_model = elastic.best_estimator_
```

[ ]:
```
best_model
```

[ ]:
```
ElasticNet(alpha=0.1, l1_ratio=0.9, random_state=999)
```

Even though we are aiming to use the elastic net only to determine the relevant variables, we can also predict the car prices and evlaute the linear model's performance. However, as we are not aiming to further improve the model performance, these values provide only an idea of the best elastic net with parameters l2 regularization alpha $= 0.1$ and l1 $= 0.9$

[ ]:
```
y_pred_elastic = elastic.predict(X_test_scaled)
mae_elastic = mean_squared_error(y_test, y_pred_elastic, squared=False)
r2_elastic  = r2_score(y_test, y_pred_elastic)

mae_elastic, r2_elastic
```

[ ]:
```
(4497.615151137075, 0.7782431149884745)
```

As to be expected, with an RMSE of 4497.61 and a R2 scor eof 77% the model performance of the elastic net is worse than our more complex, non-linear models, i.e. XGBoost and KNN. However, we can now use the elastic net results to define a new dataset containing only those features which were not set to zero.

[ ]:
```
# Supress scientific notation
np.set_printoptions(precision=4, suppress=True)

# Obtain the coefficients for the linear elastic model
coef_elastic = elastic.best_estimator_.coef_
coef_elastic
```

[ ]:
```
array([-4135.9384, -1394.5511,  5519.2434,   466.3994,   161.9258,
        627.1501,   578.9429,  2124.1885,   197.2907,   106.7519,
       -510.3685,  1044.0131,    -0.7751,  -227.0681,   -38.5964,
       -132.8247,  -219.5017,   -51.0822,  -227.6189,  1170.251 ,
       -834.8631,    14.7217,   577.1409,  -569.8666,    21.9352,
       -187.9763,     9.353 ,   -41.7956,   -19.8348,   366.0118,
        160.3434,    67.065 ,   -16.7211])
```

Since our best model is a mixture of the ridge and lasso models, by construction, we cannot

observe any coefficient being set to zero. However, we can observe the difference in magnitude of our coefficients and use a rule of thumb to drop some features. In our case, we decide to drop all coefficients with an absolute value smaller than 200, i.e. 17 features.

[ ]:
```
# Create index containing subselection of all features determined by the␣
 ↪threshold of 200
selected_features = X.columns[abs(coef_elastic) > 200]

# Define new subset
X_elastic = X[selected_features]
y_elastic = y
```

[ ]:
```
# Create new Test-Train-Split
X_tr, X_te, y_tr, y_te = train_test_split(X_elastic, y_elastic, random_state =␣
 ↪999, test_size = 0.15)
```

[ ]:
```
# Fit the final xgboost model using only the important varibles
xgb_agn.fit(X_tr,y_tr)
```

[ ]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=True, eval_metric='rmse', feature_types=None,
             gamma=0, grow_policy=None, importance_type='gain',
             interaction_constraints=None, learning_rate=0.1, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=7, max_leaves=None,
             min_child_weight=1, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=765, n_jobs=7,
             num_parallel_tree=None, random_state=999, …)
```

[ ]:
```
# Predict for test set
y_pred_xgbelastic= xgb_agn.predict(X_te)

# Compute metrics
mae_xgbelastic = mean_squared_error(y_te, y_pred_xgbelastic, squared=False)
rmse_xgbelastic = np.sqrt(mean_squared_error(y_te, y_pred_xgbelastic))
r2_xgbelastic  = r2_score(y_te, y_pred_xgbelastic)

rmse_xgbelastic, r2_xgbelastic
```

[ ]:
```
(1443.9910219365674, 0.9771418269662719)
```

As expected, the model performance slightly decreased when fitting our best xgboost model only with the most important variables, determined by the elastic net. Put differently, by following this approach we provided a computationally more efficient way of predicting car prices (when only using 17 important variables) at the expense of a lower model performance.

6.1.2 KNN

Setting the data

[ ]:
```
y, X = preprocess_Ad_Table_Trim(Ad_table, Trim).final_set(
                standardization=True,
                remove_outliers=True,
                columns_to_drop=columns_to_drop)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=123)

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.
 ↪2, random_state=123)
```

Setting the model

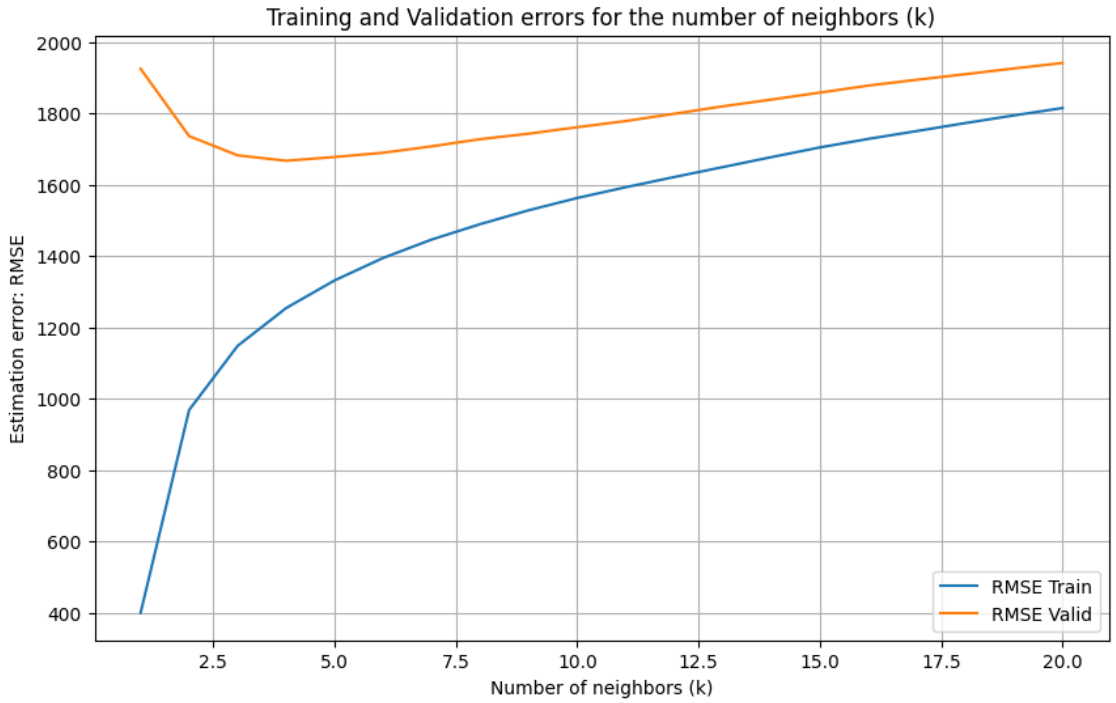
Finding the best hyperparameter

[ ]:
```
rmse = []
rmse_valid = []
number_neighbors = []

n_values = range(1, 21, 1)


for k in n_values:
    knn = KNeighborsRegressor(n_neighbors=k)

    train = knn.fit(X_train, y_train) #we train the model on the train set

    y_train_pred = train.predict(X_train) #prediction for the train set

    knn_rmse = np.sqrt(mean_squared_error(y_train, y_train_pred)) #we compute␣
 ↪the corresponding RMSE

    y_pred_valid = knn.predict(X_val) #prediction on the validation set

    knn_rmse_valid = np.sqrt(mean_squared_error(y_val, y_pred_valid)) #We␣
 ↪compute the corresponding RMSE for validation

    number_neighbors.append(k)

    print("Number of Neighbors:", k, " - RMSE (Training):", round(knn_rmse, 2),␣
 ↪" - RMSE (Validation):", round(knn_rmse_valid, 2))

    rmse.append(knn_rmse)
    rmse_valid.append(knn_rmse_valid)
```

```
Number of Neighbors: 1  - RMSE (Training): 399.65  - RMSE (Validation): 1924.84
Number of Neighbors: 2  - RMSE (Training): 968.21  - RMSE (Validation): 1736.14
Number of Neighbors: 3  - RMSE (Training): 1148.07  - RMSE (Validation): 1682.21
Number of Neighbors: 4  - RMSE (Training): 1254.05  - RMSE (Validation): 1667.01
Number of Neighbors: 5  - RMSE (Training): 1331.71  - RMSE (Validation): 1677.56
Number of Neighbors: 6  - RMSE (Training): 1394.63  - RMSE (Validation): 1689.54
Number of Neighbors: 7  - RMSE (Training): 1446.19  - RMSE (Validation): 1707.49
Number of Neighbors: 8  - RMSE (Training): 1489.39  - RMSE (Validation): 1727.61
Number of Neighbors: 9  - RMSE (Training): 1528.28  - RMSE (Validation): 1742.98
Number of Neighbors: 10  - RMSE (Training): 1562.69  - RMSE (Validation):
1761.12
Number of Neighbors: 11  - RMSE (Training): 1592.94  - RMSE (Validation):
1778.51
Number of Neighbors: 12  - RMSE (Training): 1621.47  - RMSE (Validation):
1798.84
Number of Neighbors: 13  - RMSE (Training): 1649.34  - RMSE (Validation):
1819.74
Number of Neighbors: 14  - RMSE (Training): 1677.2  - RMSE (Validation): 1838.54
Number of Neighbors: 15  - RMSE (Training): 1704.45  - RMSE (Validation):
1858.16
Number of Neighbors: 16  - RMSE (Training): 1728.37  - RMSE (Validation):
1877.82
Number of Neighbors: 17  - RMSE (Training): 1750.46  - RMSE (Validation):
1894.17
Number of Neighbors: 18  - RMSE (Training): 1772.77  - RMSE (Validation):
1909.78
Number of Neighbors: 19  - RMSE (Training): 1794.26  - RMSE (Validation):
1925.89
Number of Neighbors: 20  - RMSE (Training): 1814.87  - RMSE (Validation):
1941.13
```

[ ]:
```
#We create a dataframe with all the RMSE

rmse = np.array(rmse)
rmse_valid = np.array(rmse_valid)
number_neighbors = np.array(number_neighbors)

df_RMSE = pd.DataFrame({'RMSE_train': rmse, 'RMSE_valid': rmse_valid,␣
 ↪'number_neighbors': number_neighbors})

#We create a graph to display the results previoulsy computed for RMSE on train␣
 ↪and validation set

plt.figure(figsize=(10, 6))
plt.plot(df_RMSE['number_neighbors'].values, df_RMSE['RMSE_train'].values,␣
 ↪label='RMSE Train')
```

```
plt.plot(df_RMSE['number_neighbors'].values, df_RMSE['RMSE_valid'].values,␣
 ↪label='RMSE Valid')
plt.title('Training and Validation errors for the number of neighbors (k)')
plt.xlabel('Number of neighbors (k)')
plt.ylabel('Estimation error: RMSE')
plt.legend()
plt.grid(True)
plt.show()
```



[ ]:
```
#We display the lowest RMSE and its correponding k
lowest_RMSE = df_RMSE.idxmin()['RMSE_valid']
best_k = np.array(df_RMSE.loc[[lowest_RMSE]])

print(df_RMSE.min()['RMSE_valid'])
print(df_RMSE.loc[[lowest_RMSE]])
```

```
1667.0113560376265
   RMSE_train   RMSE_valid  number_neighbors
3  1254.05286  1667.011356                 4
```

Using the CV to find the best hyperparameter

[ ]:
```
# We create a list of k to test from 1 to 15
param_grid = {'n_neighbors': [1, 2, 3, 4, 5,6, 7, 8, 9, 10, 11, 12, 13, 14, 15]}
```

```
# Because GridSearchCV maximizes scores we take the negative RMSE
score = 'neg_root_mean_squared_error'

# We create the RMSE regressor
reg = model_selection.GridSearchCV(
neighbors.KNeighborsRegressor(),
    param_grid,
    cv=5,
    scoring=score)

# We optimize this regressor on the training set
reg.fit(X_train, y_train)

# Display the best hyperparemter on the train set
print("Best hyperparameter on the train set:")
print(reg.best_params_)

# We want a positive RMSE to compare with our previous results
rmse = -reg.best_score_

# Display the performances for each k
print("CV results :")
for mean, std, params in zip(
    reg.cv_results_['mean_test_score'],
    reg.cv_results_['std_test_score'],
    reg.cv_results_['params']
):

    print("RMSE = {:.3f} (+/- {:.03f}) for {}".format(
        (-mean),
        (std * 2),
        params
    ))
```

Applying the best model on the test set

[ ]:
```
#We train the best model on the train and predict the test set

knn = KNeighborsRegressor(n_neighbors=4)

train = knn.fit(X_train, y_train)

y_pred_train = train.predict(X_train)

y_pred_test = train.predict(X_test)

rmse_train = np.sqrt(mean_squared_error(y_train, y_pred_train))
```

```
rmse_test = np.sqrt(mean_squared_error(y_test, y_pred_test))

print('Root Mean Squared Error on the training set:', rmse_train)
print('Root Mean Squared Error on the test set:', rmse_test)
```

```
Root Mean Squared Error on the training set: 1254.052859722444
Root Mean Squared Error on the validation set: 1719.3488597363346
```

Reduction method: PCA

[ ]:
```
# We use the PCA function to compute the components and the corresponding␣
↪variance
pca = PCA()
X_pca = pca.fit_transform(X_train)
```

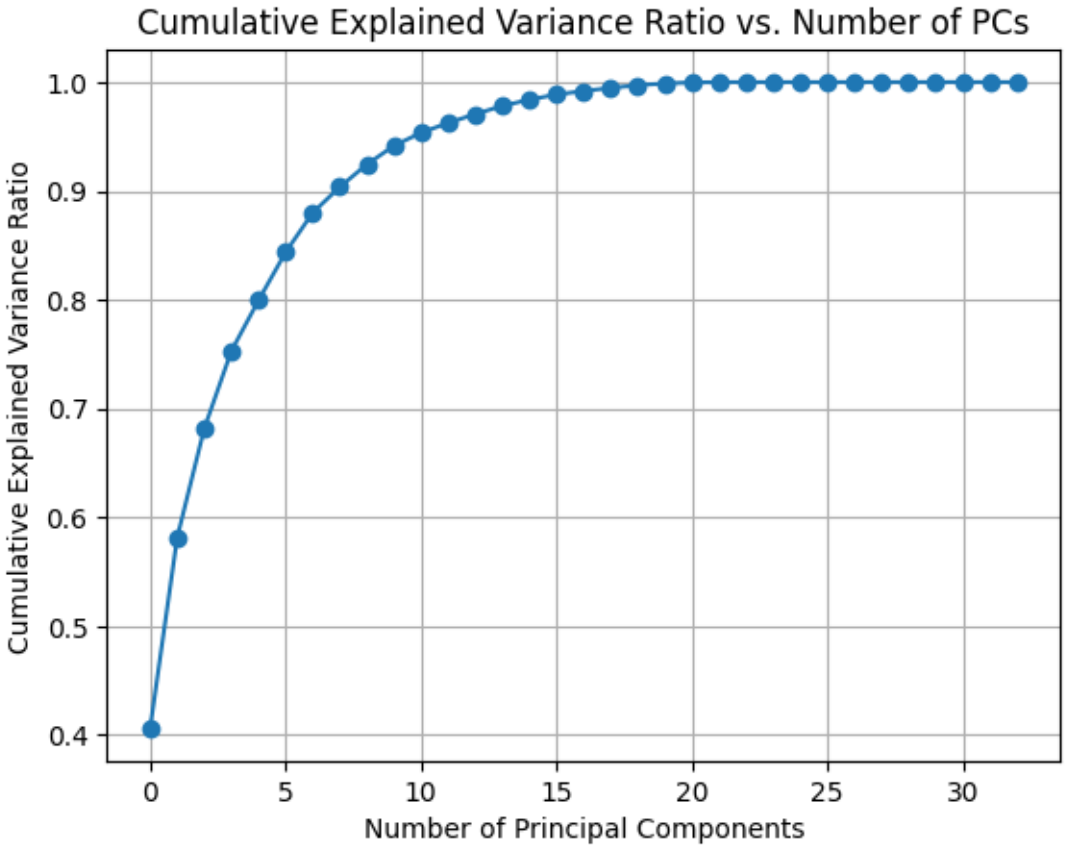
[ ]:
```
cumulative_variance_ratio = pca.explained_variance_ratio_.cumsum()
df_cum_var = pd.DataFrame({
    'Principal Components': range(1, len(cumulative_variance_ratio) + 1),
    'Cumulative Variance Ratio': cumulative_variance_ratio
})
df_cum_var
```

[ ]:

	Principal Components	Cumulative Variance Ratio
0	1	0.406173
1	2	0.581577
2	3	0.681716
3	4	0.752933
4	5	0.799350
5	6	0.843540
6	7	0.879896
7	8	0.903567
8	9	0.924325
9	10	0.941438
10	11	0.953442
11	12	0.962466
12	13	0.970534
13	14	0.978110
14	15	0.983817
15	16	0.988540
16	17	0.991820
17	18	0.994579
18	19	0.997200
19	20	0.998701
20	21	0.999710
21	22	0.999907
22	23	0.999950
23	24	0.999970
24	25	0.999983

25	26	0.999990
26	27	0.999997
27	28	0.999999
28	29	0.999999
29	30	1.000000
30	31	1.000000
31	32	1.000000
32	33	1.000000

[ ]:
```
# We plot the principal components analysis with their respective orders
cumulative_variance_ratio = pca.explained_variance_ratio_.cumsum()
plt.plot(cumulative_variance_ratio, marker='o')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.title('Cumulative Explained Variance Ratio vs. Number of PCs')
plt.grid(True)
plt.show()
```



We select n = 11 (environ 95% of variance explained)

[ ]:
```
# We train the KNN model on the reduced dataset with 11 components

knn = KNeighborsRegressor(n_neighbors=4)
n_components = 11
pca = PCA(n_components=n_components)
X_train_pca = pca.fit_transform(X_train)
X_val_pca = pca.transform(X_val)
X_test_pca = pca.transform(X_test)

train_pca = knn.fit(X_train_pca, y_train)

y_pred_train_pca = train_pca.predict(X_train_pca)
y_pred_test_pca = train_pca.predict(X_test_pca)

rmse_train_pca = np.sqrt(mean_squared_error(y_train, y_pred_train_pca))
print(f"Mean Squared Error with PCA for the train set: {rmse_train_pca}")
rmse_test_pca = np.sqrt(mean_squared_error(y_test, y_pred_test_pca))
print(f"Mean Squared Error with PCA for the test set: {rmse_test_pca}")
```

```
Mean Squared Error with PCA for the train set: 1272.464842621307
Mean Squared Error with PCA for the test set: 1735.8583858783857
```

6.1.3 Neural network

Setting data

[ ]:
```
# Call the preprocessing class
y, X = preprocess_classical_data.final_set(standardization=True,␣
 ↪remove_outliers=True, columns_to_drop=columns_to_drop)

# Using pytorch, we need the data to be tensors
X_tensor = torch.tensor(X.values, dtype=torch.float32)
y_tensor = torch.tensor(y.values, dtype=torch.float32).view(-1, 1)
dataset = TensorDataset(X_tensor, y_tensor)
```

Setting the model

[ ]:
```
# Using pytorch, we create a class for the multi-layer perceptron model
# This class will allow us to have a flexible model so we can swap parameters␣
 ↪to tune them


class MLP_regression(torch.nn.Module):

    def __init__(self, num_features, num_layers, layer_sizes, func_name,␣
 ↪dropout_probas):

        # Create a Module List with pytorch
        super(MLP_regression, self).__init__()
```

```
        self.module = torch.nn.ModuleList()

        for i in range(num_layers):
            # All of the following function will be added to the pytorh module,␣
↪which is the model

            # Input layer (first layer)
            if i == 0:
                self.module.append(torch.nn.Linear(num_features,␣
↪layer_sizes[i]))
                # Activation function
                self.module.append(self.Get_activation_function(func_name))
                # Dropout probabilities
                self.module.append(torch.nn.Dropout(dropout_probas[i]))

            # Output layer (last layer)
            elif i == num_layers - 1:
                self.module.append(torch.nn.Linear(layer_sizes[i-1], 1))

            # Hidden layers
            else:
                self.module.append(torch.nn.Linear(layer_sizes[i-1],␣
↪layer_sizes[i]))
                # Activation function
                self.module.append(self.Get_activation_function(func_name))
                # Dropout probabilities
                self.module.append(torch.nn.Dropout(dropout_probas[i]))

    # function to call the activation function according to the name we set up␣
↪as input
    def Get_activation_function(self, func_name):

        if func_name == 'Relu' :
          self.FF = torch.nn.ReLU()
        elif func_name == 'Tanh':
          self.FF =  torch.nn.Tanh()
        elif func_name == 'Sigmoid':
          self.FF =  torch.nn.Sigmoid()
        return self.FF

    # Function to pass the input and compute the predicted values
    def forward(self, x):
        for module in self.module:
            x = module(x)
        return x
```

Hand-made functions

[ ]:
```
# Function to train the model
# Takes an iterator as input (dataloader)
def train(model, iterator, optimizer, criterion):

    epoch_loss = 0

    # Pytorch defines that the model is training
    model.train()

    # Call x and y while looping in the dataloader (also according to the batch size)
    for (x, y) in tqdm(iterator, desc="Training", leave=False):

        # Reset gradient to zerp
        optimizer.zero_grad()

        # Put x's in the model and predict y
        y_pred = model(x)

        # compute the loss
        mse = criterion(y_pred, y)
        # We want the RMSE, not the MSE, so we take the squared root of MSE
        loss = torch.sqrt(mse)

        loss.backward()
        optimizer.step()

        epoch_loss += loss.item()
    return epoch_loss / len(iterator)


# Function to evaluate the model
# Same as train but the weights aren't updated
def evaluate(model, iterator, criterion):
    epoch_loss = 0
    model.eval()
    with torch.no_grad():
        for (x, y) in tqdm(iterator, desc="Evaluating", leave=False):

            y_pred = model(x)

            mse = criterion(y_pred, y)
            loss = torch.sqrt(mse)

            epoch_loss += loss.item()
    return epoch_loss / len(iterator)
```

```
# Just a displayer as the process is long
def epoch_time(start_time, end_time):
    elapsed_time = end_time - start_time
    elapsed_mins = int(elapsed_time / 60)
    elapsed_secs = int(elapsed_time - (elapsed_mins * 60))
    return elapsed_mins, elapsed_secs
```

Splitting data

[ ]:
```
# Split folders into train and test
train_size = int(0.7 * (len(dataset)))
eval_size = int(0.1 * len(dataset))
test_size = len(dataset) - train_size - eval_size

# The csv is splitted
train_dataset, test_dataset, eval_dataset = random_split(dataset, [train_size,␣
 ↪test_size, eval_size])
```

[ ]:
```
train_size
```

[ ]:
```
141325
```

Hyperparameters optimization

[ ]:
```
def wandb_train(config=None):
  with wandb.init(config=config):

    config=wandb.config

    # Parameters to swap
    BATCH_SIZE = config.batch_size
    learning_rate = config.learning_rate
    EPOCHS = config.epoch
    activation_function = config.activation_func
    Dropout_probas=config.Dropout_probas


    # Setup loaders
    test_loader = DataLoader(test_dataset, batch_size=BATCH_SIZE, shuffle=False)
    train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE,␣
 ↪shuffle=True)
    valid_loader = DataLoader(eval_dataset, batch_size=BATCH_SIZE,␣
 ↪shuffle=False)

    # fixed parameters

    # Number of input features
```

```
    num_features = len(X.columns)
    # Total number of layers
    num_layers = 5
    # Sizes of each layer
    layer_sizes = [30, 25, 20, 10]


    # Setup the model
    model = MLP_regression(num_features,
                           num_layers,
                           layer_sizes,
                           activation_function,
                           Dropout_probas)

    # Setup criterion and the optimizer
    criterion = torch.nn.MSELoss()
    optimizer = optim.Adam(model.parameters(), lr=learning_rate)

    for epoch in trange(EPOCHS, desc="Epochs"):

        start_time = time.monotonic()

        train_loss = train(model, train_loader, optimizer, criterion)
        valid_loss = evaluate(model, valid_loader, criterion)


        end_time = time.monotonic()

        epoch_mins, epoch_secs = epoch_time(start_time, end_time)

        wandb.log({"valid loss": valid_loss, "train loss": train_loss, "epoch":␣
↪epoch})
```

[ ]:
```
# Connect to the library Wandb to run parameters swap
%%capture
!pip install wandb --upgrade
import wandb
wandb.login()
```

[ ]:
```
# Setup wandb config

sweep_config = {
    'method': 'random'
    }

metric = {
  'name': 'valid loss',
```

```
  'goal': 'minimize'
}

sweep_config['metric'] = metric

# Store all the parameters to swap in a dictionary

parameters_dict = {

    'learning_rate': {
        'distribution' : 'uniform',
        'min' : 5e-4,
        'max' : 0.07,
    },
    'batch_size' : {
        'distribution': 'q_log_uniform_values',
        'q': 8,
        'min': 30,
        'max': 200,
    },
    'epoch' : {
        'distribution' : 'q_log_uniform_values',
        'min' : 3,
        'max' : 15,
    },

    'Dropout_probas': {
        'values' : [[0.1, 0.1, 0.1, 0.1],
         [0.3, 0.2, 0.1, 0.1],
          [0.1, 0.1, 0.2, 0.3],
           [0.1, 0.4, 0.4, 0.1],
            [0.3, 0.1, 0.1, 0.3]],
    },
    "activation_func":{
        'values' : ['Tanh','Sigmoid', 'Relu'],
    },
    }


sweep_config['parameters'] = parameters_dict
```

[ ]:
```
# Connect to the project
sweep_id = wandb.sweep(sweep_config, project="Car_prices_prediction")
```

```
Create sweep with ID: 4fdy4jzx
Sweep URL: https://wandb.ai/ojlt/Car_prices_prediction/sweeps/4fdy4jzx
```

[ ]:
```
# Run wandb
start_time = time.time()
wandb.agent("i5sf60ub", wandb_train, count=30)
elapsed = (time.time() - start_time)/60
print(f'Total Time: {elapsed:.2f} min')
```

```
wandb: Agent Starting Run: ehtvj7i1 with config:
wandb:     Dropout_probas: [0.1, 0.1, 0.1, 0.1]
wandb:     activation_func: Sigmoid
wandb:     batch_size: 64
wandb:     epoch: 6
wandb:     learning_rate: 0.04769333045815452
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Epochs:   0%|          | 0/6 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/2209 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/2209 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/2209 [00:00<?, ?it/s]
```

```
Traceback (most recent call last):
  File "<ipython-input-27-442746d577b8>", line 45, in wandb_train
    train_loss = train(model, train_loader, optimizer, criterion)
  File "<ipython-input-13-6d43ded40bdb>", line 15, in train
    optimizer.step()
  File "/usr/local/lib/python3.10/dist-packages/torch/optim/optimizer.py", line
373, in wrapper
    out = func(*args, **kwargs)
  File "/usr/local/lib/python3.10/dist-packages/torch/optim/optimizer.py", line
76, in _use_grad
    ret = func(self, *args, **kwargs)
  File "/usr/local/lib/python3.10/dist-packages/torch/optim/adam.py", line 163,
in step
    adam(
  File "/usr/local/lib/python3.10/dist-packages/torch/optim/adam.py", line 311,
in adam
    func(params,
```

```
  File "/usr/local/lib/python3.10/dist-packages/torch/optim/adam.py", line 384,
in _single_tensor_adam
    exp_avg.lerp_(grad, 1 - beta1)
Exception
```

```
VBox(children=(Label(value='0.001 MB of 0.001 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
wandb: Agent Starting Run: 39t4rqdd with config:
wandb:      Dropout_probas: [0.1, 0.1, 0.2, 0.3]
wandb:      activation_func: Sigmoid
wandb:      batch_size: 48
wandb:      epoch: 8
wandb:      learning_rate: 0.03601913151885313
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/8 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: tfa2wbqc with config:
wandb:      Dropout_probas: [0.1, 0.1, 0.1, 0.1]
wandb:      activation_func: Relu
wandb:      batch_size: 64
wandb:      epoch: 11
wandb:      learning_rate: 0.05274439535004306

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/11 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: x19by73f with config:
wandb:     Dropout_probas: [0.1, 0.4, 0.4, 0.1]
wandb:     activation_func: Relu
wandb:     batch_size: 144
wandb:     epoch: 7
wandb:     learning_rate: 0.0012690152560986105

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/7 [00:00<?, ?it/s]

Training:   0%|          | 0/982 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/141 [00:00<?, ?it/s]

Training:   0%|          | 0/982 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/141 [00:00<?, ?it/s]

Training:   0%|          | 0/982 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/141 [00:00<?, ?it/s]

Training:   0%|          | 0/982 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/141 [00:00<?, ?it/s]

Training:   0%|          | 0/982 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/141 [00:00<?, ?it/s]

Training:   0%|          | 0/982 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/141 [00:00<?, ?it/s]

Training:   0%|          | 0/982 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/141 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: awiemhlg with config:
wandb:     Dropout_probas: [0.1, 0.1, 0.2, 0.3]
wandb:     activation_func: Sigmoid
wandb:     batch_size: 32
wandb:     epoch: 4
wandb:     learning_rate: 0.0027504696382678953

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/4 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]
```

```
VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),
↪FloatProgress(value=1.0, max=1.0)))
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: bbvz4wnm with config:
wandb:     Dropout_probas: [0.3, 0.2, 0.1, 0.1]
wandb:     activation_func: Tanh
wandb:     batch_size: 32
wandb:     epoch: 13
wandb:     learning_rate: 0.0496286358425851

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/13 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: wcy80qae with config:
wandb:     Dropout_probas: [0.1, 0.1, 0.2, 0.3]
wandb:     activation_func: Relu
wandb:     batch_size: 192
wandb:     epoch: 11
wandb:     learning_rate: 0.05909285549276058

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/11 [00:00<?, ?it/s]

Training:   0%|          | 0/737 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/106 [00:00<?, ?it/s]

Training:   0%|          | 0/737 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/106 [00:00<?, ?it/s]

Training:   0%|          | 0/737 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/106 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/737 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/106 [00:00<?, ?it/s]

Training:   0%|          | 0/737 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/106 [00:00<?, ?it/s]

Training:   0%|          | 0/737 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/106 [00:00<?, ?it/s]

Training:   0%|          | 0/737 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/106 [00:00<?, ?it/s]

Training:   0%|          | 0/737 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/106 [00:00<?, ?it/s]

Training:   0%|          | 0/737 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/106 [00:00<?, ?it/s]

Training:   0%|          | 0/737 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/106 [00:00<?, ?it/s]

Training:   0%|          | 0/737 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/106 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: 86fldcyc with config:
wandb:     Dropout_probas: [0.1, 0.1, 0.2, 0.3]
wandb:     activation_func: Sigmoid
wandb:     batch_size: 80
wandb:     epoch: 5
wandb:     learning_rate: 0.03968921189908074

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/5 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/1767 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]

Training:   0%|          | 0/1767 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]

Training:   0%|          | 0/1767 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]

Training:   0%|          | 0/1767 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]

Training:   0%|          | 0/1767 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: h9y4vn0i with config:
wandb:     Dropout_probas: [0.1, 0.1, 0.1, 0.1]
wandb:     activation_func: Sigmoid
wandb:     batch_size: 32
wandb:     epoch: 3
wandb:     learning_rate: 0.01935425761981169

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/3 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Training:   0%|          | 0/4417 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]
```

```
VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: 5u0256b6 with config:
wandb:     Dropout_probas: [0.3, 0.1, 0.1, 0.3]
wandb:     activation_func: Tanh
wandb:     batch_size: 120
wandb:     epoch: 15
wandb:     learning_rate: 0.02129557365615059

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/15 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: ddnbzec5 with config:
wandb:     Dropout_probas: [0.3, 0.1, 0.1, 0.3]
wandb:     activation_func: Relu
wandb:     batch_size: 160
wandb:     epoch: 13
wandb:     learning_rate: 0.06084459529687838

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/13 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/884 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: pgzp2vom with config:
wandb:      Dropout_probas: [0.3, 0.2, 0.1, 0.1]
wandb:      activation_func: Tanh
wandb:      batch_size: 80
```

```
wandb:     epoch: 14
wandb:     learning_rate: 0.04775990231297315
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
Epochs:   0%|          | 0/14 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]
Training:   0%|          | 0/1767 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]

Training:   0%|          | 0/1767 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]

Training:   0%|          | 0/1767 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/253 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: asgtnwyc with config:
wandb:     Dropout_probas: [0.1, 0.1, 0.2, 0.3]
wandb:     activation_func: Sigmoid
wandb:     batch_size: 120
wandb:     epoch: 10
wandb:     learning_rate: 0.04594664449741485

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/10 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

Training:   0%|          | 0/1178 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/169 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: yhqqri3g with config:
wandb:     Dropout_probas: [0.3, 0.2, 0.1, 0.1]
wandb:     activation_func: Sigmoid
wandb:     batch_size: 48
wandb:     epoch: 8
wandb:     learning_rate: 0.008426659324875258

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/8 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'), FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: ifbeeg7u with config:
wandb:     Dropout_probas: [0.1, 0.1, 0.2, 0.3]
wandb:     activation_func: Relu
wandb:     batch_size: 48
wandb:     epoch: 14
wandb:     learning_rate: 0.010894623432767037

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/14 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

Training:   0%|          | 0/2945 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/421 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Sweep Agent: Waiting for job.
wandb: Job received.
wandb: Agent Starting Run: 1dcffdk8 with config:
wandb:     Dropout_probas: [0.1, 0.1, 0.1, 0.1]
wandb:     activation_func: Sigmoid
wandb:     batch_size: 40
wandb:     epoch: 11
wandb:     learning_rate: 0.01457337416571102
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/11 [00:00<?, ?it/s]

Training:   0%|          | 0/3534 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

Training:   0%|          | 0/3534 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

Training:   0%|          | 0/3534 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

Training:   0%|          | 0/3534 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

Training:   0%|          | 0/3534 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

Training:   0%|          | 0/3534 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

Training:   0%|          | 0/3534 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

Training:   0%|          | 0/3534 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

Training:   0%|          | 0/3534 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

Training:   0%|          | 0/3534 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

Training:   0%|          | 0/3534 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: 301p1ml3 with config:
wandb:      Dropout_probas: [0.1, 0.1, 0.2, 0.3]
wandb:      activation_func: Relu
wandb:      batch_size: 72
wandb:      epoch: 11
wandb:      learning_rate: 0.005761977374277701

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/11 [00:00<?, ?it/s]

Training:   0%|          | 0/1963 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/281 [00:00<?, ?it/s]

Training:   0%|          | 0/1963 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/281 [00:00<?, ?it/s]

Training:   0%|          | 0/1963 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/281 [00:00<?, ?it/s]

Training:   0%|          | 0/1963 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/281 [00:00<?, ?it/s]

Training:   0%|          | 0/1963 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/281 [00:00<?, ?it/s]

Training:   0%|          | 0/1963 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/281 [00:00<?, ?it/s]

Training:   0%|          | 0/1963 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/281 [00:00<?, ?it/s]

Training:   0%|          | 0/1963 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/281 [00:00<?, ?it/s]

Training:   0%|          | 0/1963 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/281 [00:00<?, ?it/s]

Training:   0%|          | 0/1963 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/281 [00:00<?, ?it/s]

Training:   0%|          | 0/1963 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/281 [00:00<?, ?it/s]

VBox(children=(Label(value='0.010 MB of 0.010 MB uploaded\r'),␣
  ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: gw6coadv with config:
wandb:     Dropout_probas: [0.3, 0.1, 0.1, 0.3]
wandb:     activation_func: Sigmoid
wandb:     batch_size: 128
wandb:     epoch: 3
wandb:     learning_rate: 0.0464981680904098

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/3 [00:00<?, ?it/s]

Training:   0%|          | 0/1105 [00:00<?, ?it/s]
```

Evaluation of the model

[ ]:
```python
best_dropout_probas = [0.1,0.1,0.1,0.1]
best_BATCH_SIZE = 64
best_learning_rate = 0.05274
best_activation_function = "Relu"

# Number of input features
num_features = len(X.columns)
# Total number of layers
num_layers = 5
# Sizes of each layer
layer_sizes = [30, 25, 20, 10]

loss_eval_1 = []
loss_train_1 = []
```

```
model = MLP_regression(num_features,
                           num_layers,
                           layer_sizes,
                           best_activation_function,
                           best_dropout_probas)

# Setup loaders
test_loader = DataLoader(test_dataset, batch_size=best_BATCH_SIZE,␣
 ↪shuffle=False)
train_loader = DataLoader(train_dataset, batch_size=best_BATCH_SIZE,␣
 ↪shuffle=True)
valid_loader = DataLoader(eval_dataset, batch_size=best_BATCH_SIZE,␣
 ↪shuffle=False)

criterion = torch.nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=best_learning_rate)
best_valid_loss = float('inf')

EPOCHS = 20
for epoch in trange(EPOCHS, desc="Epochs"):

    start_time = time.monotonic()

    train_loss = train(model, train_loader, optimizer, criterion)
    valid_loss = evaluate(model, valid_loader, criterion)

    loss_eval_1.append(valid_loss)
    loss_train_1.append(train_loss)


    if valid_loss < best_valid_loss:
        best_valid_loss = valid_loss
        torch.save(model, '/content/drive/MyDrive/ML BigProject/model_MLP.pt')
        torch.save(model.state_dict(), '/content/drive/MyDrive/ML BigProject/
 ↪model_MLP_weights.pt')
        print("---- New best loss ----")
        print("Epoch : ", epoch+1)

    end_time = time.monotonic()

    epoch_mins, epoch_secs = epoch_time(start_time, end_time)

    print(f'Epoch: {epoch+1:02} | Epoch Time: {epoch_mins}m {epoch_secs}s')
    print(f'\tTrain Loss: {train_loss:.3f}')
    print(f'\t Val. Loss: {valid_loss:.3f}')
```

```
Epochs:   0%|          | 0/20 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  1
Epoch: 01 | Epoch Time: 0m 5s
        Train Loss: 4361.493
         Val. Loss: 3052.031

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  2
Epoch: 02 | Epoch Time: 0m 6s
        Train Loss: 4145.891
         Val. Loss: 2897.364

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 03 | Epoch Time: 0m 6s
        Train Loss: 4081.977
         Val. Loss: 3119.710

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 04 | Epoch Time: 0m 6s
        Train Loss: 4024.947
         Val. Loss: 2975.727

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 05 | Epoch Time: 0m 6s
        Train Loss: 4017.481
         Val. Loss: 3474.377

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 06 | Epoch Time: 0m 6s
        Train Loss: 3961.230
         Val. Loss: 3206.516

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]
```

```
Epoch: 07 | Epoch Time: 0m 6s
        Train Loss: 3977.672
         Val. Loss: 3115.154

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 08 | Epoch Time: 0m 6s
        Train Loss: 3957.357
         Val. Loss: 3396.692

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 09 | Epoch Time: 0m 5s
        Train Loss: 3971.717
         Val. Loss: 2897.836

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  10
Epoch: 10 | Epoch Time: 0m 6s
        Train Loss: 3892.050
         Val. Loss: 2706.537

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  11
Epoch: 11 | Epoch Time: 0m 5s
        Train Loss: 3824.617
         Val. Loss: 2703.197

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 12 | Epoch Time: 0m 5s
        Train Loss: 3869.360
         Val. Loss: 3010.415

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 13 | Epoch Time: 0m 6s
        Train Loss: 3779.531
         Val. Loss: 2771.910

Training:   0%|          | 0/2209 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 14 | Epoch Time: 0m 5s
        Train Loss: 3774.658
         Val. Loss: 2720.595

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 15 | Epoch Time: 0m 5s
        Train Loss: 3782.583
         Val. Loss: 3567.217

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 16 | Epoch Time: 0m 5s
        Train Loss: 3793.796
         Val. Loss: 2829.682

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 17 | Epoch Time: 0m 6s
        Train Loss: 3743.300
         Val. Loss: 2813.898

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 18 | Epoch Time: 0m 5s
        Train Loss: 3720.657
         Val. Loss: 4562.335

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 19 | Epoch Time: 0m 6s
        Train Loss: 3710.818
         Val. Loss: 2901.095

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 20 | Epoch Time: 0m 5s
        Train Loss: 3742.283
         Val. Loss: 3310.058
```

[ ]:
```python
# Test the model
best_model = torch.load('/content/drive/MyDrive/ML BigProject/model_MLP.pt')
test_loss = evaluate(best_model, test_loader, criterion)
```

```
print(f"Test Loss: {test_loss:.3f}")
```

```
Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]
```

```
Test Loss: 2688.819
```

[ ]:
```
# try to train and eval the model without dropout probabilities
best_dropout_probas = [0,0,0,0]
best_BATCH_SIZE = 64
best_learning_rate = 0.05274
best_activation_function = "Relu"

# Number of input features
num_features = len(X.columns)
# Total number of layers
num_layers = 5
# Sizes of each layer
layer_sizes = [30, 25, 20, 10]


loss_eval_2 = []
loss_train_2 = []


model = MLP_regression(num_features,
                           num_layers,
                           layer_sizes,
                           best_activation_function,
                           best_dropout_probas)

# Setup loaders

test_loader = DataLoader(test_dataset, batch_size=best_BATCH_SIZE,␣
 ↪shuffle=False)
train_loader = DataLoader(train_dataset, batch_size=best_BATCH_SIZE,␣
 ↪shuffle=True)
valid_loader = DataLoader(eval_dataset, batch_size=best_BATCH_SIZE,␣
 ↪shuffle=False)

criterion = torch.nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=best_learning_rate)
best_valid_loss = float('inf')

EPOCHS = 30
for epoch in trange(EPOCHS, desc="Epochs"):

    start_time = time.monotonic()
```

```
    train_loss = train(model, train_loader, optimizer, criterion)
    valid_loss = evaluate(model, valid_loader, criterion)

    loss_eval_2.append(valid_loss)
    loss_train_2.append(train_loss)

    if valid_loss < best_valid_loss:
        best_valid_loss = valid_loss
        torch.save(model, '/content/drive/MyDrive/ML BigProject/
↪model_MLP_no_dropout.pt')
        print("---- New best loss ----")
        print("Epoch : ", epoch+1)

    end_time = time.monotonic()

    epoch_mins, epoch_secs = epoch_time(start_time, end_time)

    print(f'Epoch: {epoch+1:02} | Epoch Time: {epoch_mins}m {epoch_secs}s')
    print(f'\tTrain Loss: {train_loss:.3f}')
    print(f'\t Val. Loss: {valid_loss:.3f}')
```

```
Epochs:   0%|          | 0/30 [00:00<?, ?it/s]

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  1
Epoch: 01 | Epoch Time: 0m 5s
        Train Loss: 3241.611
         Val. Loss: 2920.699

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 02 | Epoch Time: 0m 5s
        Train Loss: 2879.573
         Val. Loss: 3282.056

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  3
Epoch: 03 | Epoch Time: 0m 5s
        Train Loss: 2661.652
         Val. Loss: 2575.002

Training:   0%|          | 0/2209 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  4
Epoch: 04 | Epoch Time: 0m 5s
        Train Loss: 2591.965
         Val. Loss: 2466.934

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  5
Epoch: 05 | Epoch Time: 0m 5s
        Train Loss: 2493.100
         Val. Loss: 2415.466

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 06 | Epoch Time: 0m 5s
        Train Loss: 2434.685
         Val. Loss: 2433.948

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  7
Epoch: 07 | Epoch Time: 0m 5s
        Train Loss: 2373.903
         Val. Loss: 2405.849

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 08 | Epoch Time: 0m 5s
        Train Loss: 2329.345
         Val. Loss: 2407.375

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  9
Epoch: 09 | Epoch Time: 0m 5s
        Train Loss: 2289.023
         Val. Loss: 2337.019

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]
```

```
---- New best loss ----
Epoch :  10
Epoch: 10 | Epoch Time: 0m 5s
        Train Loss: 2271.863
         Val. Loss: 2201.334

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  11
Epoch: 11 | Epoch Time: 0m 5s
        Train Loss: 2231.482
         Val. Loss: 2200.097

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  12
Epoch: 12 | Epoch Time: 0m 5s
        Train Loss: 2223.866
         Val. Loss: 2195.716

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  13
Epoch: 13 | Epoch Time: 0m 5s
        Train Loss: 2201.443
         Val. Loss: 2155.440

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 14 | Epoch Time: 0m 5s
        Train Loss: 2187.788
         Val. Loss: 2364.361

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 15 | Epoch Time: 0m 5s
        Train Loss: 2169.562
         Val. Loss: 2358.995

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]
```

```
Epoch: 16 | Epoch Time: 0m 5s
        Train Loss: 2171.321
         Val. Loss: 2162.160

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  17
Epoch: 17 | Epoch Time: 0m 5s
        Train Loss: 2156.006
         Val. Loss: 2087.124

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 18 | Epoch Time: 0m 5s
        Train Loss: 2146.334
         Val. Loss: 2181.278

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 19 | Epoch Time: 0m 5s
        Train Loss: 2138.270
         Val. Loss: 2219.009

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 20 | Epoch Time: 0m 5s
        Train Loss: 2126.691
         Val. Loss: 2170.147

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 21 | Epoch Time: 0m 5s
        Train Loss: 2122.389
         Val. Loss: 2096.409

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 22 | Epoch Time: 0m 5s
        Train Loss: 2107.730
         Val. Loss: 2149.951

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]
```

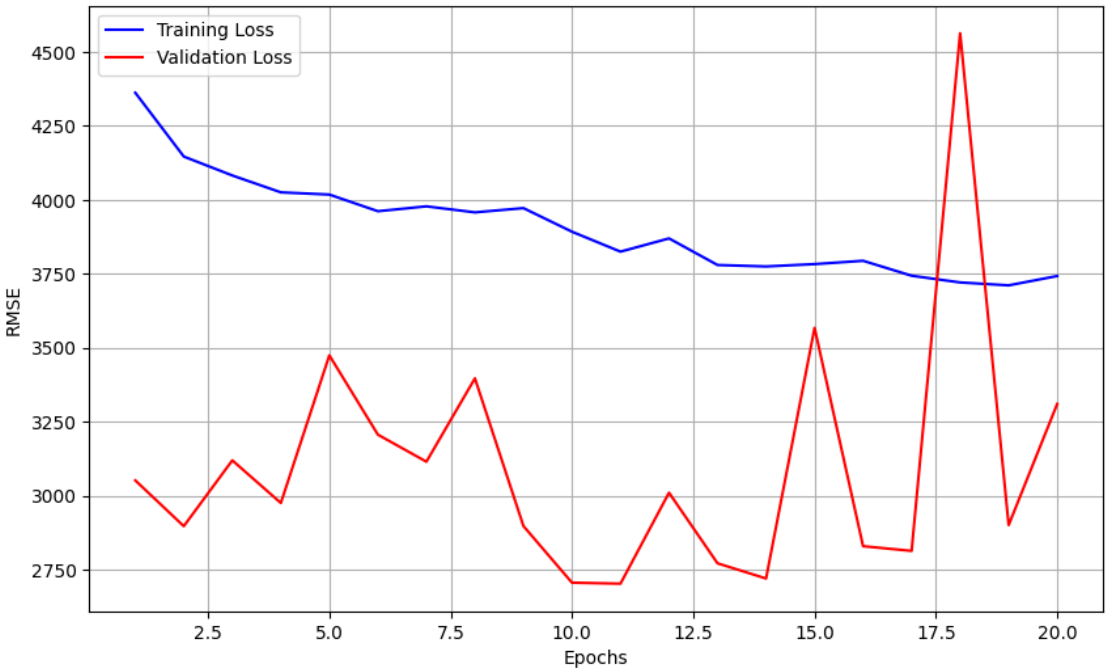
```
Epoch: 23 | Epoch Time: 0m 5s
        Train Loss: 2098.909
         Val. Loss: 2127.133

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 24 | Epoch Time: 0m 5s
        Train Loss: 2101.403
         Val. Loss: 2110.767

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

---- New best loss ----
Epoch :  25
Epoch: 25 | Epoch Time: 0m 5s
        Train Loss: 2097.105
         Val. Loss: 2068.568

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 26 | Epoch Time: 0m 5s
        Train Loss: 2081.862
         Val. Loss: 2099.350

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 27 | Epoch Time: 0m 5s
        Train Loss: 2084.457
         Val. Loss: 2127.125

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 28 | Epoch Time: 0m 5s
        Train Loss: 2078.929
         Val. Loss: 2132.559

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]

Epoch: 29 | Epoch Time: 0m 5s
        Train Loss: 2071.765
         Val. Loss: 2101.926

Training:   0%|          | 0/2209 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/316 [00:00<?, ?it/s]
```

```
---- New best loss ----
Epoch :  30
Epoch: 30 | Epoch Time: 0m 5s
        Train Loss: 2077.846
         Val. Loss: 2061.793
```

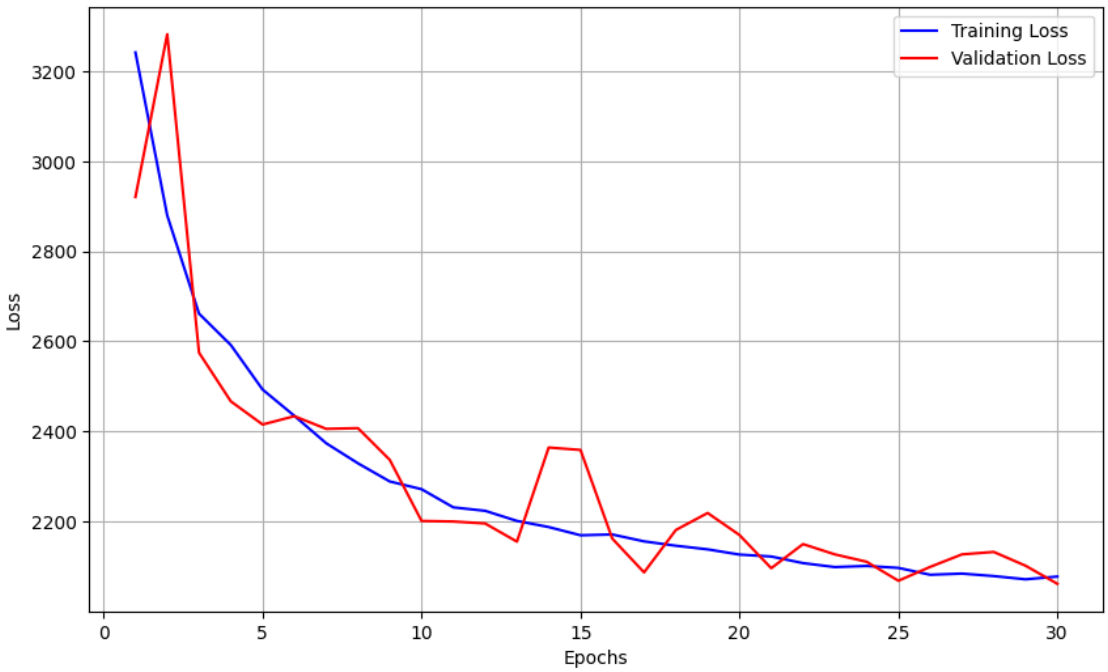
[ ]:
```python
# Test the model without dropout probabilities
best_model = torch.load('/content/drive/MyDrive/ML BigProject/
 ↪model_MLP_no_dropout.pt')
test_loss = evaluate(best_model, test_loader, criterion)
print(f"Test Loss: {test_loss:.3f}")
```

```
Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]

Test Loss: 2057.508
```

[ ]:
```python
epochs = range(1, len(loss_eval_1) + 1)

# Plotting the training and validation loss
plt.figure(figsize=(10, 6))
plt.plot(epochs, loss_train_1, 'b-', label='Training Loss')
plt.plot(epochs, loss_eval_1, 'r-', label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('RMSE')
plt.legend()
plt.grid(True)
plt.show()
```



[ ]:
```
epochs = range(1, len(loss_eval_2) + 1)

# Plotting the training and validation loss
plt.figure(figsize=(10, 6))
plt.plot(epochs, loss_train_2, 'b-', label='Training Loss')
plt.plot(epochs, loss_eval_2, 'r-', label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()
```



[ ]:
```
# Show the best model
criterion = torch.nn.MSELoss()
optimizer = optim.Adam(best_model.parameters(), lr=best_learning_rate)
print('-------------------- \n',
      'optimizer : ', optimizer, '\n -------------------- \n',
      'criterion : ', criterion, '\n -------------------- \n',
      'model structure : ', best_model, '\n --------------------')
```

```
--------------------
 optimizer :  Adam (
Parameter Group 0
    amsgrad: False
```

```
    betas: (0.9, 0.999)
    capturable: False
    differentiable: False
    eps: 1e-08
    foreach: None
    fused: None
    lr: 0.05274
    maximize: False
    weight_decay: 0
)
 --------------------
 criterion :  MSELoss()
 --------------------
 model structure :  MLP_regression(
  (module): ModuleList(
    (0): Linear(in_features=33, out_features=30, bias=True)
    (1): ReLU()
    (2): Dropout(p=0, inplace=False)
    (3): Linear(in_features=30, out_features=25, bias=True)
    (4): ReLU()
    (5): Dropout(p=0, inplace=False)
    (6): Linear(in_features=25, out_features=20, bias=True)
    (7): ReLU()
    (8): Dropout(p=0, inplace=False)
    (9): Linear(in_features=20, out_features=10, bias=True)
    (10): ReLU()
    (11): Dropout(p=0, inplace=False)
    (12): Linear(in_features=10, out_features=1, bias=True)
  )
  (FF): ReLU()
)
 --------------------
```

Optimisation of the number of layers (Could not run...) for some reasons, Wandb stopped to work in our code 2 weeks before the deadline. We tried to run it several times without success

[ ]:
```
sweep_config = {
    'method': 'random'
    }

metric = {
  'name': 'valid loss',
  'goal': 'minimize'
}

sweep_config['metric'] = metric
```

```
parameters_dict = {

    'learning_rate': {
        'distribution' : 'uniform',
        'min' : 2e-2,
        'max' : 0.1,
    },

    "layer_size":{
        'values' : [[30, 25, 20, 10],
                    [30, 25, 20, 15, 10, 5],
                    [30, 28, 25, 20, 5],
                    [20, 15, 8, 5],
                    [25, 20, 5],
                    [28, 20, 18, 10, 5],
                    [10, 15, 20, 25, 30]],
    },
    }


sweep_config['parameters'] = parameters_dict
```

[ ]:
```
best_dropout_probas = [0.1,0.1,0.1,0.1]
best_BATCH_SIZE = 64
best_learning_rate = 0.05274

# Number of input features
num_features = len(X.columns)
# Total number of layers
num_layers = 5
# Sizes of each layer
layer_sizes = [30, 25, 20, 10]
```

[ ]:
```
def wandb_train(config=None):
  with wandb.init(config=config):

    config=wandb.config

    ####### Parameters to swap
    layer_sizes = config.layer_size
    learning_rate = config.learning_rate




    ####### fixed parameters
```

```
    # Number of input features
    num_features = len(X.columns)
    # Total number of layers
    num_layers = len(layer_sizes) + 1
    # Batch size
    BATCH_SIZE = 64
    # Dropout probabilities
    Dropout_probas=[0.1,0.1,0.1,0.1]
    # Activation functions
    activation_function = "Relu"



    # Setup loaders
    test_loader = DataLoader(test_dataset, batch_size=BATCH_SIZE, shuffle=False)
    train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE,␣
↪shuffle=True)
    valid_loader = DataLoader(eval_dataset, batch_size=BATCH_SIZE,␣
↪shuffle=False)


    # Setup the model
    model = MLP_regression(num_features,
                           num_layers,
                           layer_sizes,
                           activation_function,
                           Dropout_probas)

    # Setup criterion and the optimizer
    criterion = torch.nn.MSELoss()
    optimizer = optim.Adam(model.parameters(), lr=learning_rate)

    EPOCHS = 30
    for epoch in trange(EPOCHS, desc="Epochs"):

        start_time = time.monotonic()

        train_loss = train(model, train_loader, optimizer, criterion)
        valid_loss = evaluate(model, valid_loader, criterion)


        end_time = time.monotonic()

        epoch_mins, epoch_secs = epoch_time(start_time, end_time)
```

```
        wandb.log({"valid loss": valid_loss, "train loss": train_loss, "epoch":␣
↪epoch})
```

[ ]:
```
sweep_id = wandb.sweep(sweep_config, project="Car_prices_prediction")
```

```
Create sweep with ID: 4wjogn8s
Sweep URL: https://wandb.ai/ojlt/Car_prices_prediction/sweeps/4wjogn8s
```

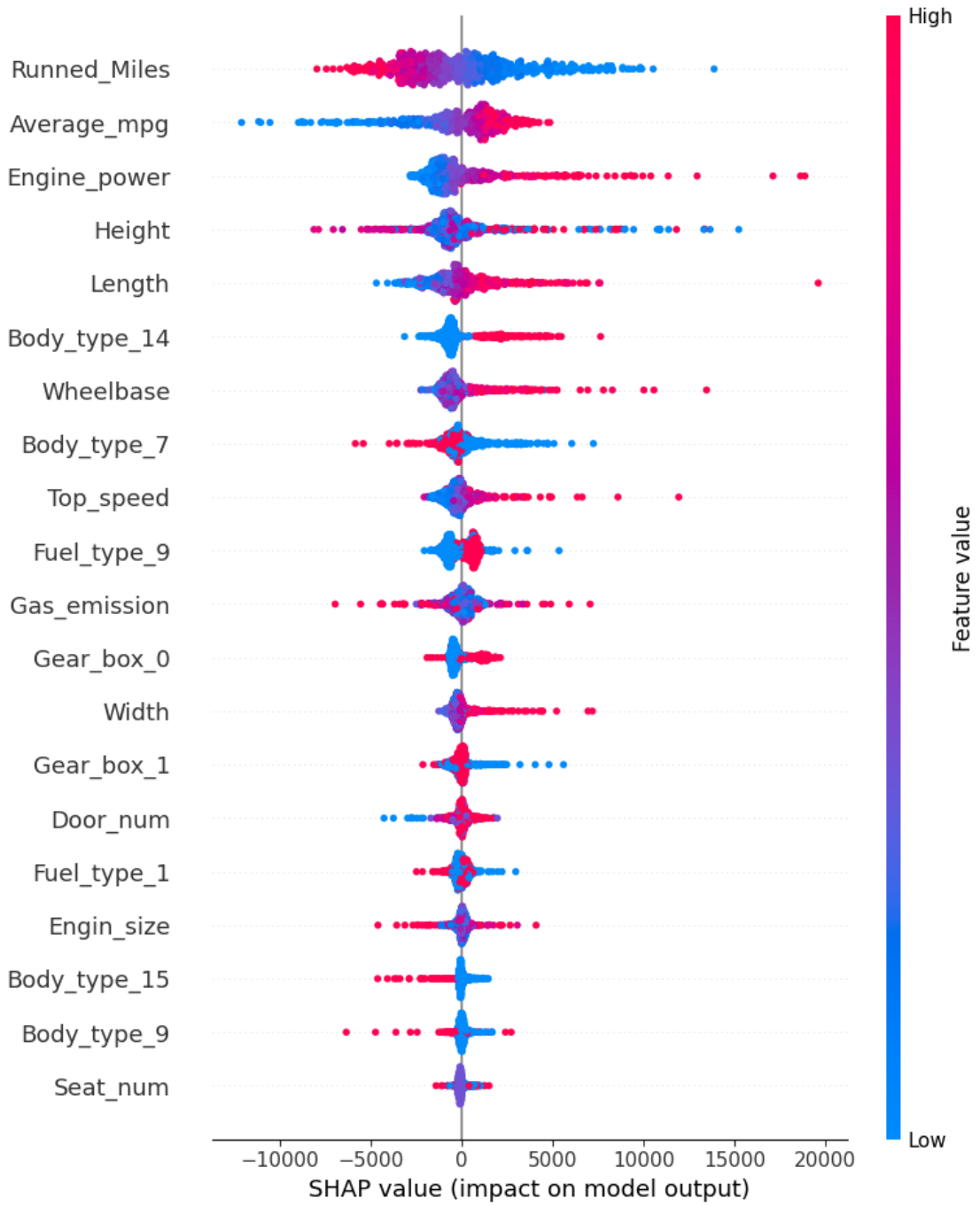
[ ]:
```
start_time = time.time()
wandb.agent("4wjogn8s", wandb_train, count=60)
elapsed = (time.time() - start_time)/60
print(f'Total Time: {elapsed:.2f} min')
```

```
wandb: Agent Starting Run: qfuce83g with config:
wandb:     layer_size: [30, 25, 20, 15, 10, 5]
wandb:     learning_rate: 0.03102890708245161

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Traceback (most recent call last):
  File "<ipython-input-16-28bd73191b67>", line 36, in wandb_train
    model = MLP_regression(num_features,
  File "<ipython-input-9-159f5465e35c>", line 35, in __init__
    self.module.append(torch.nn.Dropout(dropout_probas[i]))
IndexError: list index out of range

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Run qfuce83g errored: IndexError('list index out of range')
wandb: ERROR Run qfuce83g errored: IndexError('list
index out of range')
wandb: Ctrl + C detected. Stopping sweep.

Total Time: 0.32 min
```

Agnostic methods

[ ]:
```
background_batches = []
num_background_batches = 10  # Set this to the desired number of batches
```

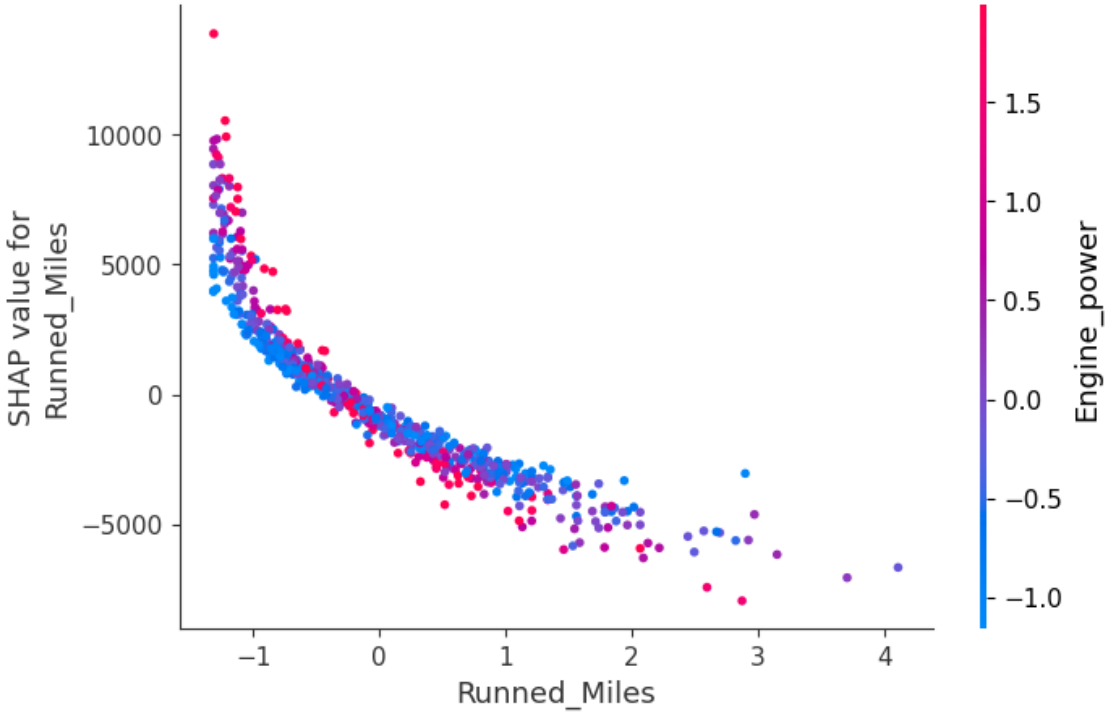
```
for i, data in enumerate(test_loader):
    if i >= num_background_batches:
        break
    if isinstance(data, list) or isinstance(data, tuple):
        # Assuming the first element is the input data
        background_batches.append(data[0])
    else:
        background_batches.append(data)

# Concatenate all the collected batches
background_data = torch.cat(background_batches, dim=0)

# Initialize the SHAP explainer with more background data
explainer = shap.GradientExplainer(best_model, background_data)

# Load multiple batches for test data
test_batches = []
num_test_batches = 10

for i, data in enumerate(test_loader):
    if i >= num_test_batches:
        break
    if isinstance(data, list) or isinstance(data, tuple):
        test_batches.append(data[0])
    else:
        test_batches.append(data)

# Concatenate all the collected test data
test_data = torch.cat(test_batches, dim=0)

# Compute SHAP values
shap_values = explainer.shap_values(test_data)

# Generate the summary plot
shap.summary_plot(shap_values, test_data, feature_names=X.columns)
```



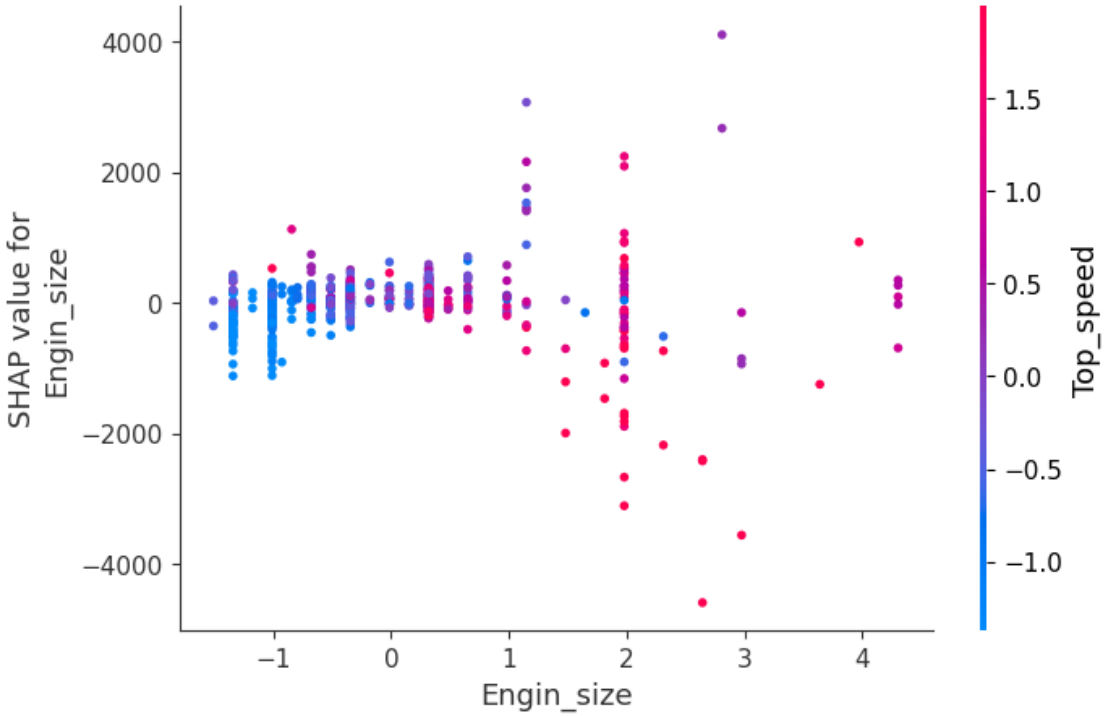
[ ]:
```
# Dependence plot

columns = X.columns
test_data_numpy = test_data.cpu().numpy() if isinstance(test_data, torch.
↪Tensor) else test_data
```

```
for col in range(len(columns)):
  print(columns[col])
  shap.dependence_plot(columns[col], shap_values, test_data_numpy,␣
 ↪feature_names=X.columns)
```

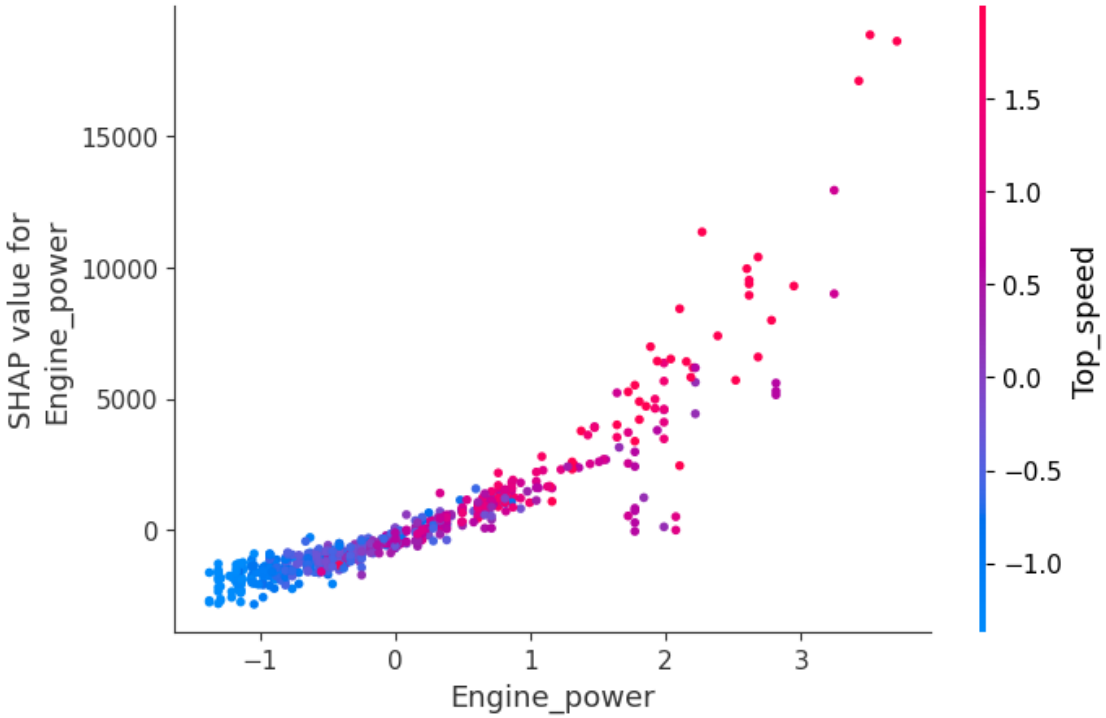
```
Runned_Miles
```



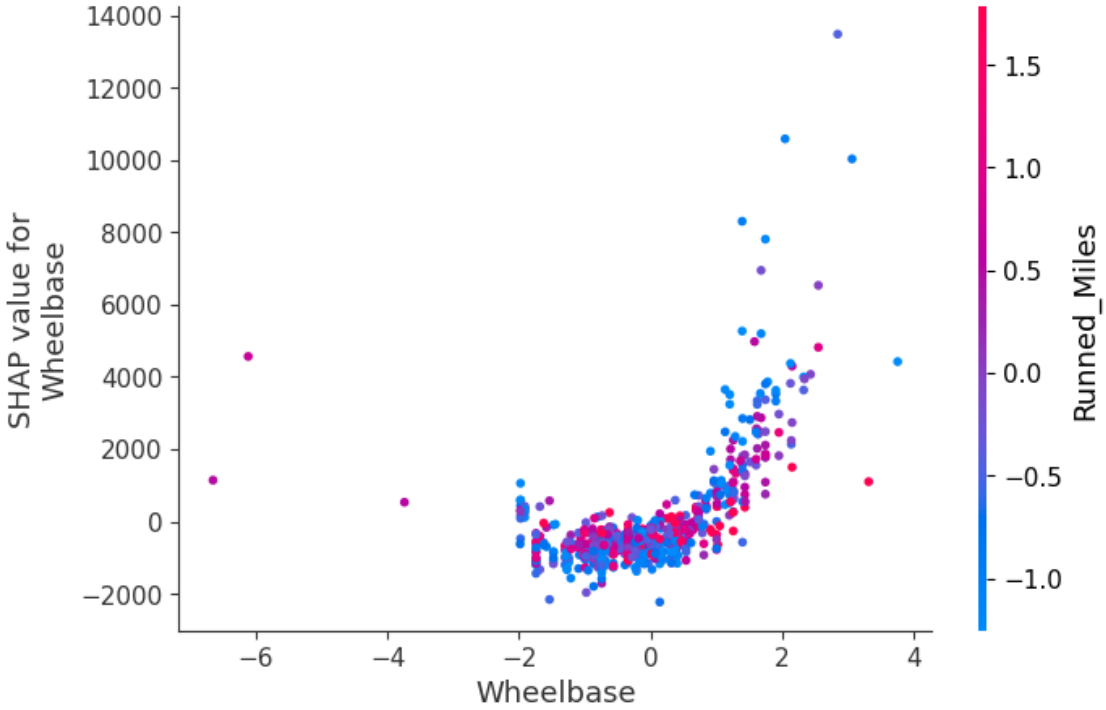

```
Engin_size
```



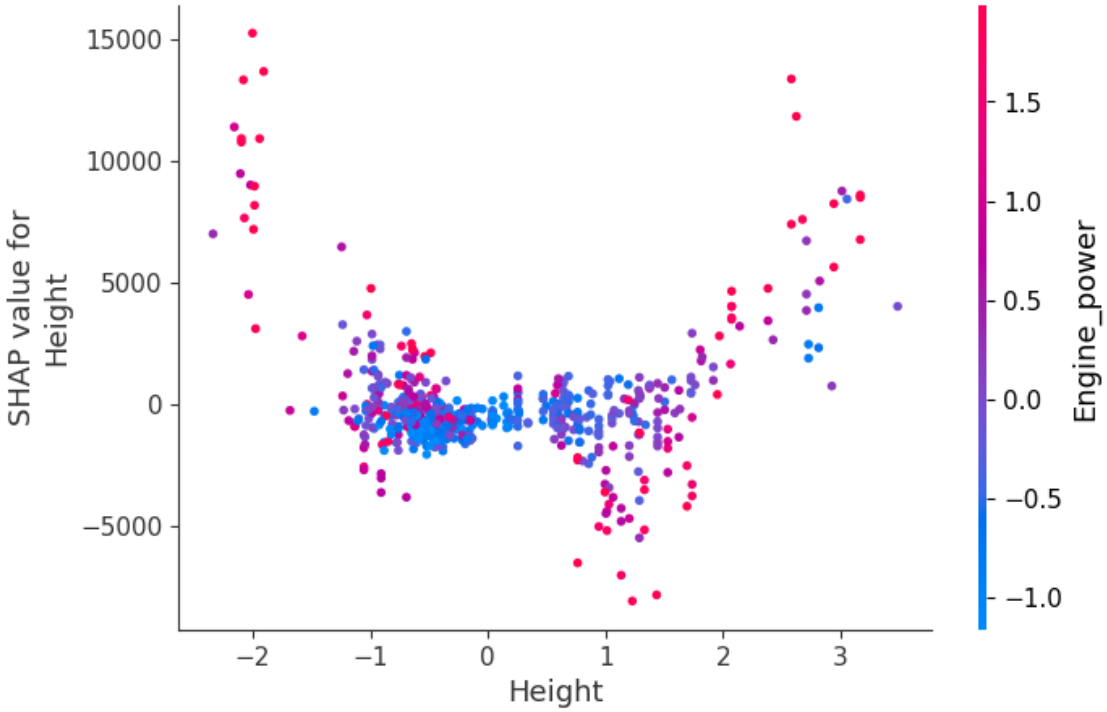
`Engine_power`



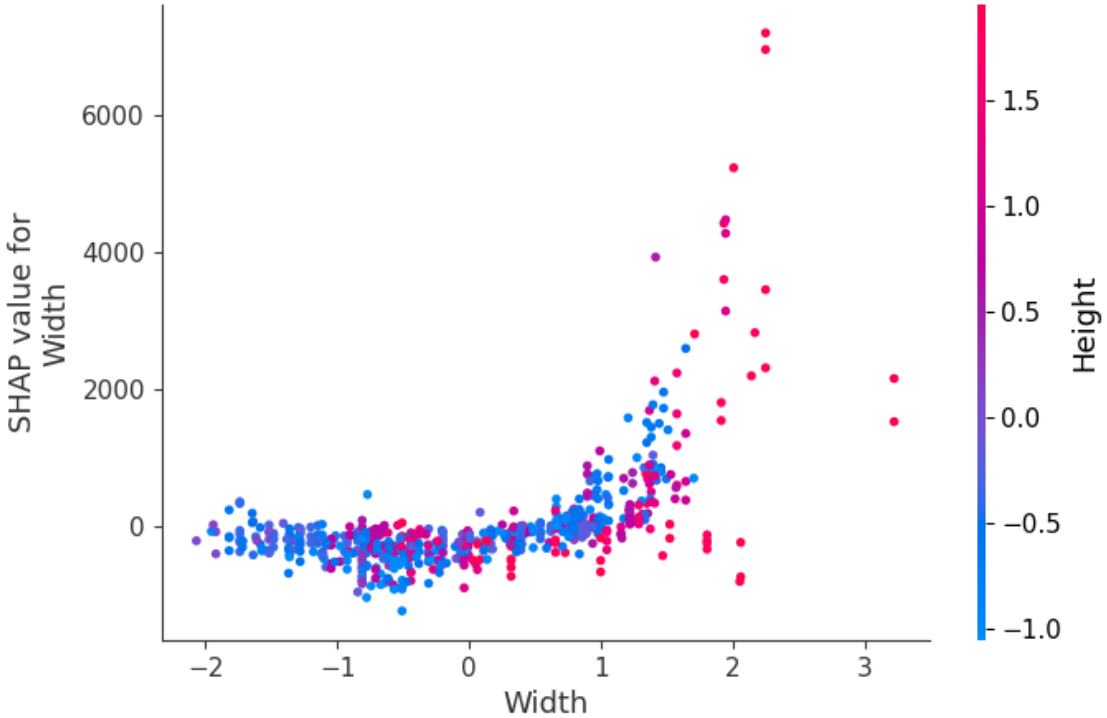
Wheelbase



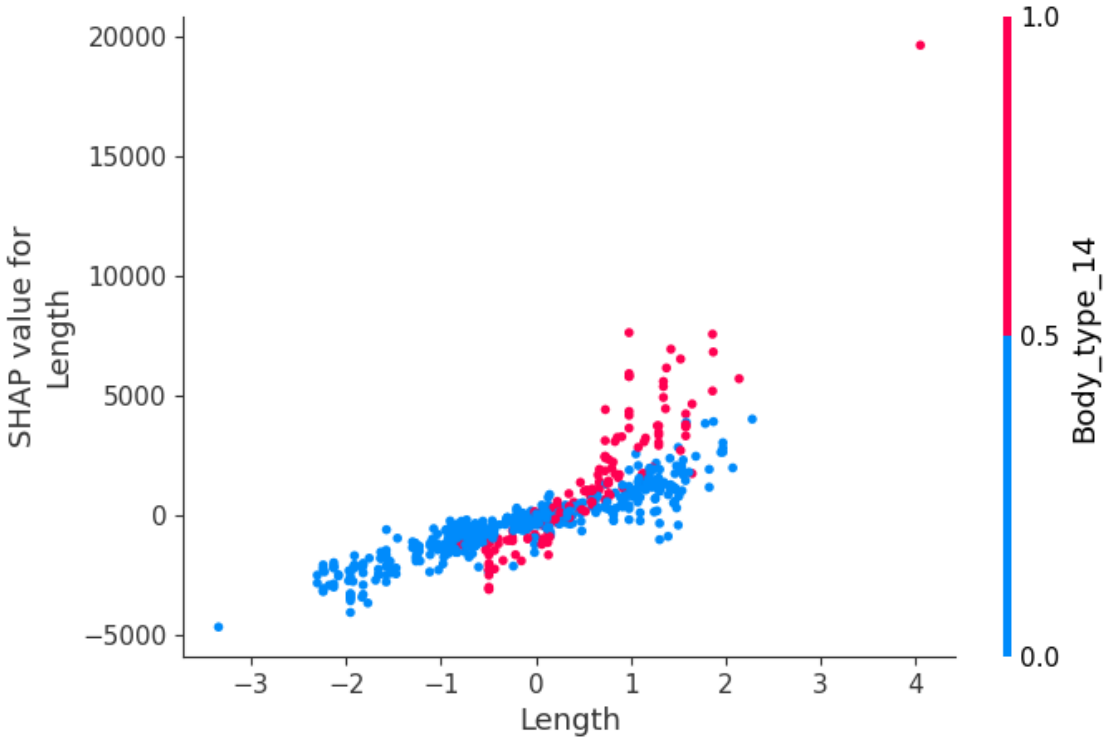
Height


SHAP value for Height
Height
Engine_power
15000
10000
5000
0
−5000
−2
−1
0
1
2
3
1.5
1.0
0.5
0.0
−0.5
−1.0

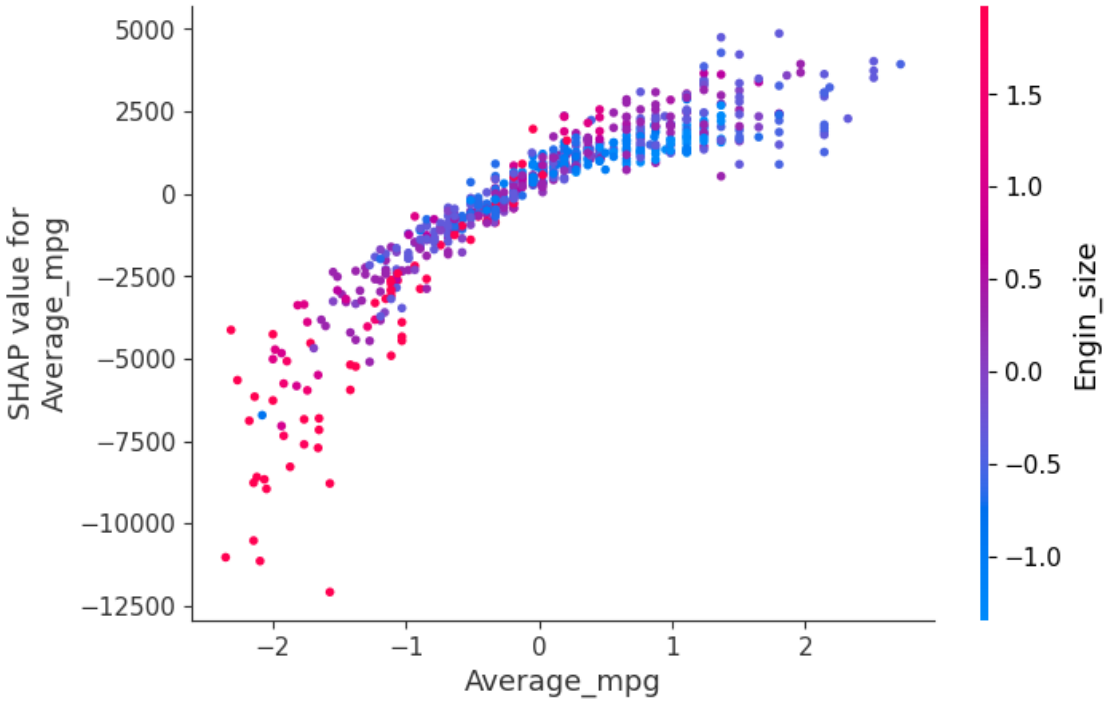
Width


SHAP value for Width
Width
Height
6000
4000
2000
0
−2
−1
0
1
2
3
1.5
1.0
0.5
0.0
−0.5
−1.0

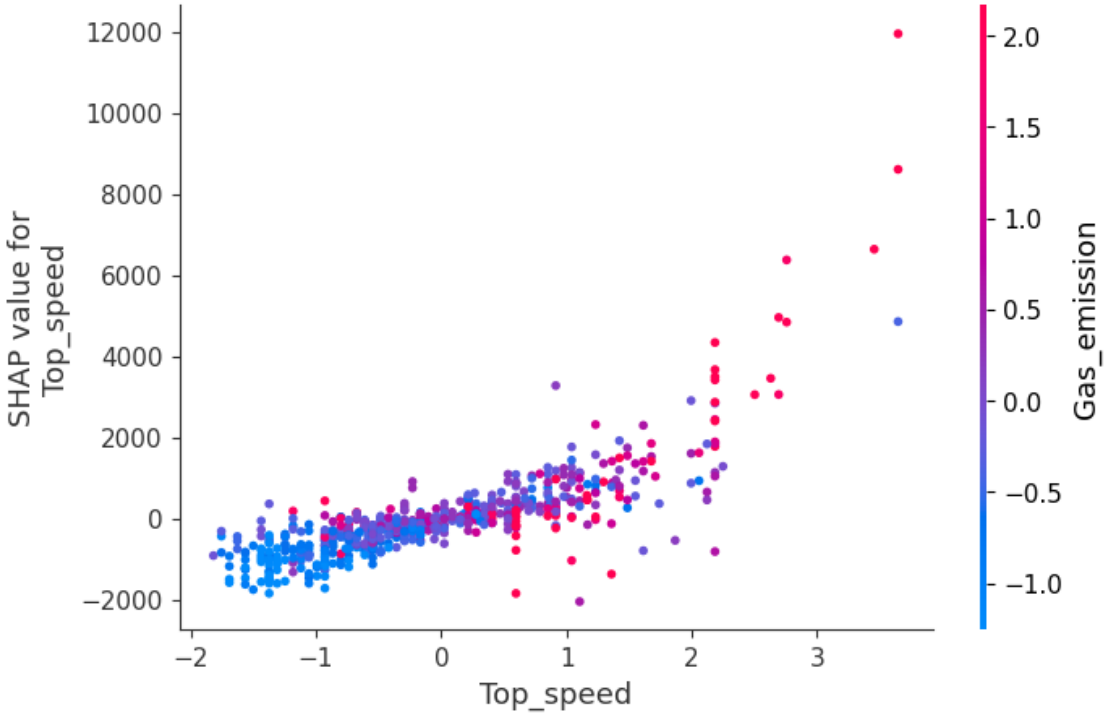
Length



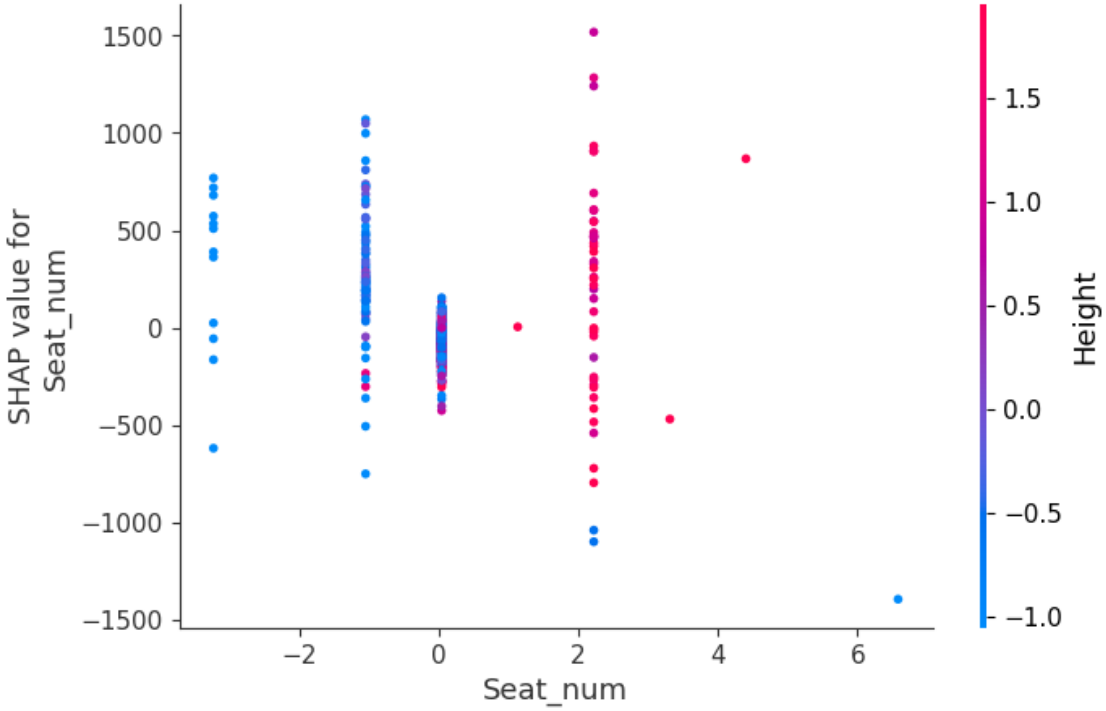
Average_mpg


SHAP value for
Average_mpg
5000
2500
0
−2500
−5000
−7500
−10000
−12500
−2
−1
0
1
2
Average_mpg
1.5
1.0
0.5
0.0
−0.5
−1.0
Engin_size

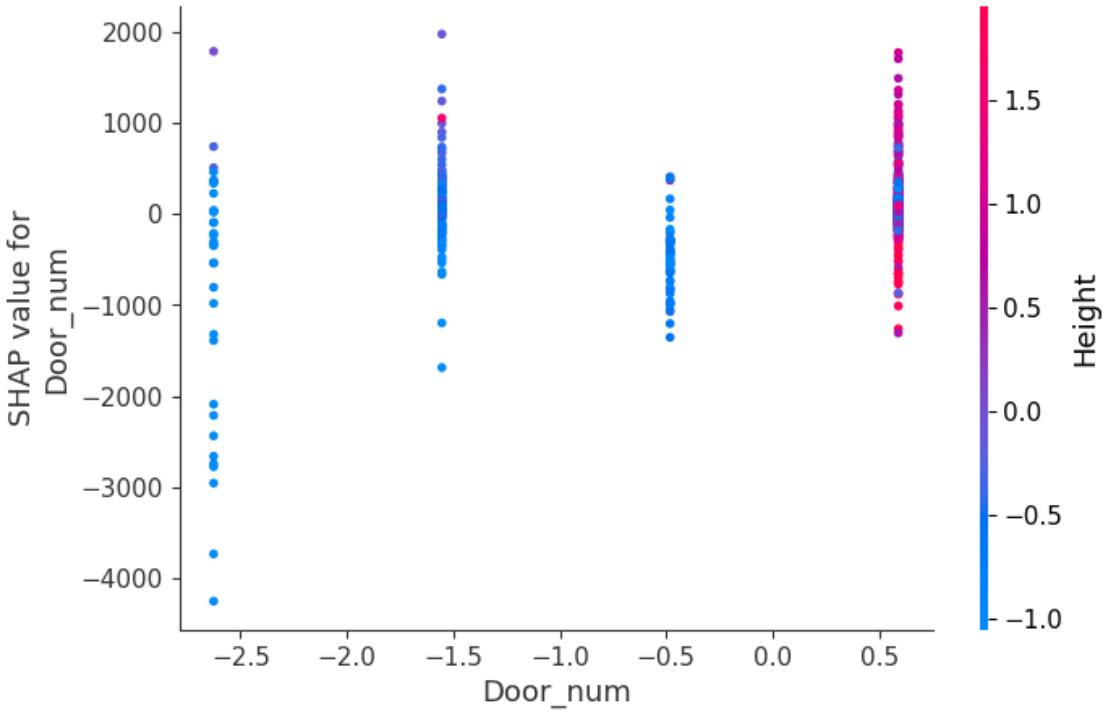
Top_speed


SHAP value for
Top_speed
12000
10000
8000
6000
4000
2000
0
−2000
−2
−1
0
1
2
3
Top_speed
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
Gas_emission

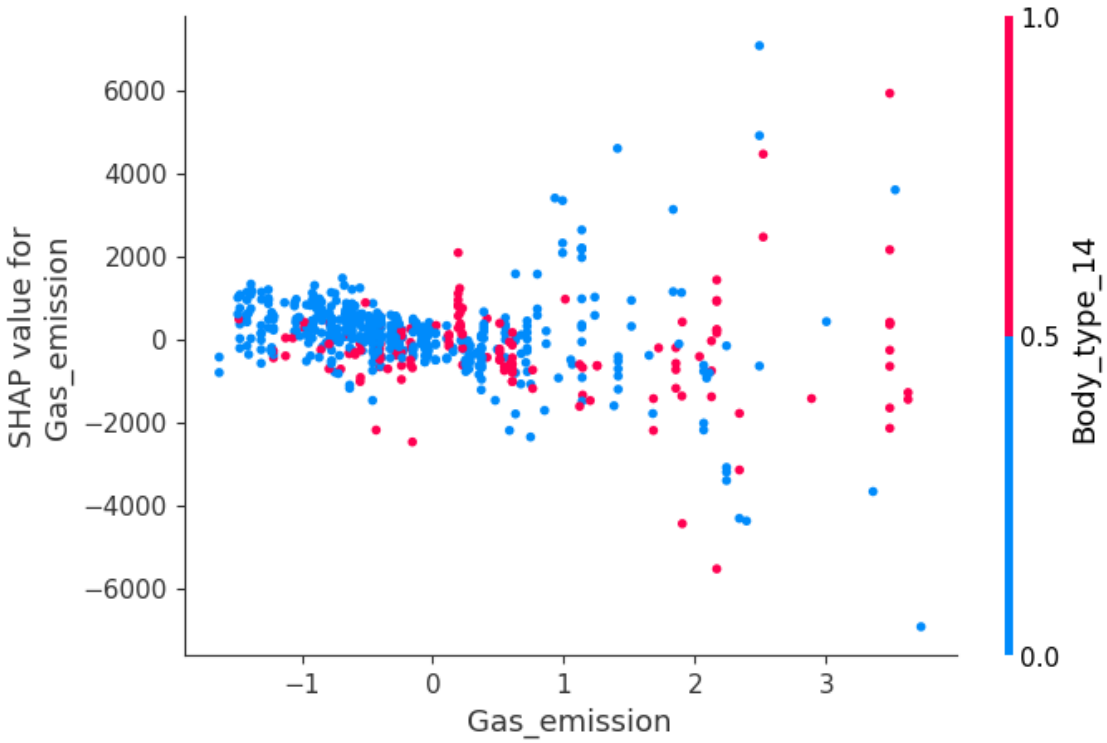
Seat_num



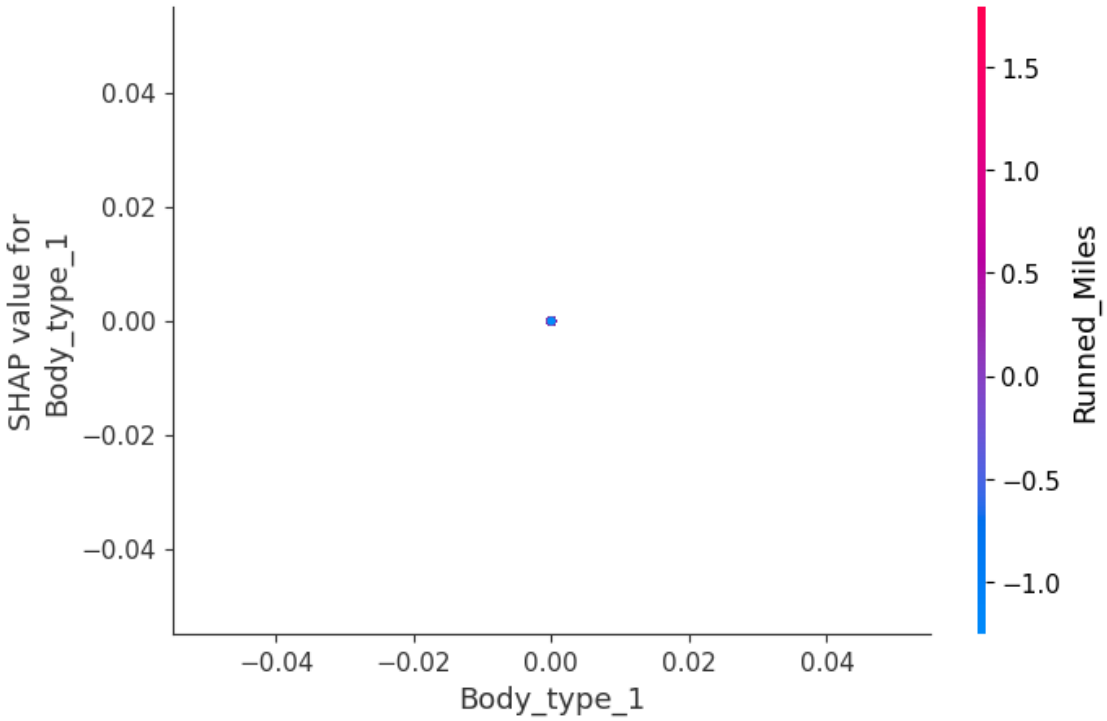
Door_num



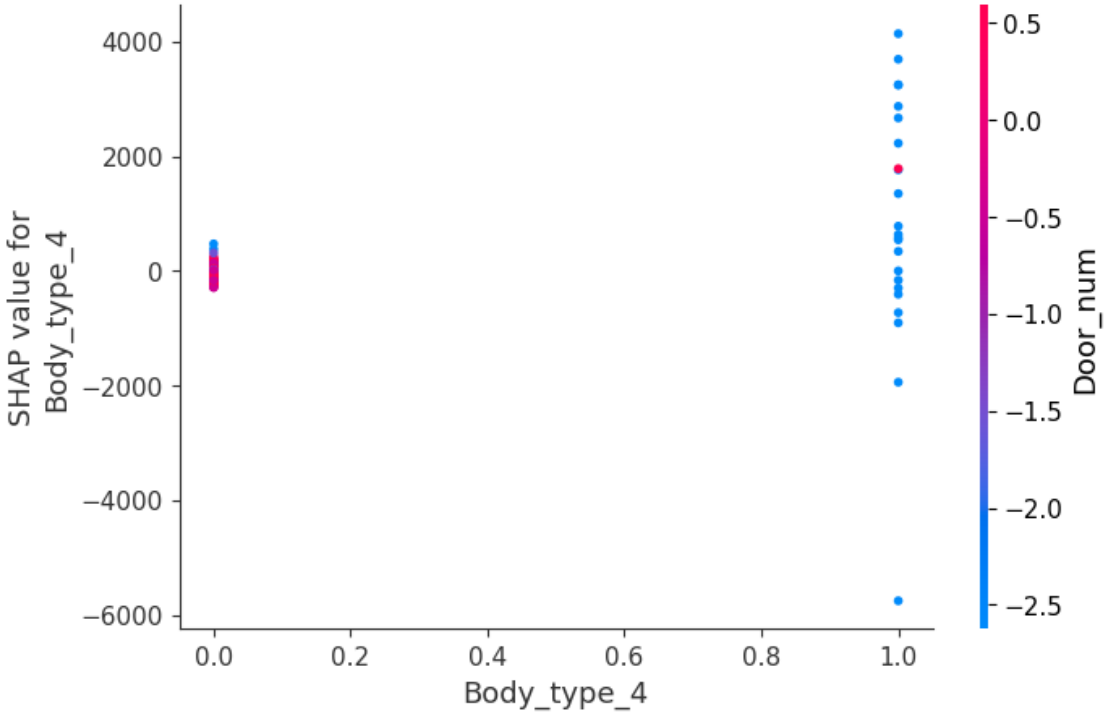
Gas_emission



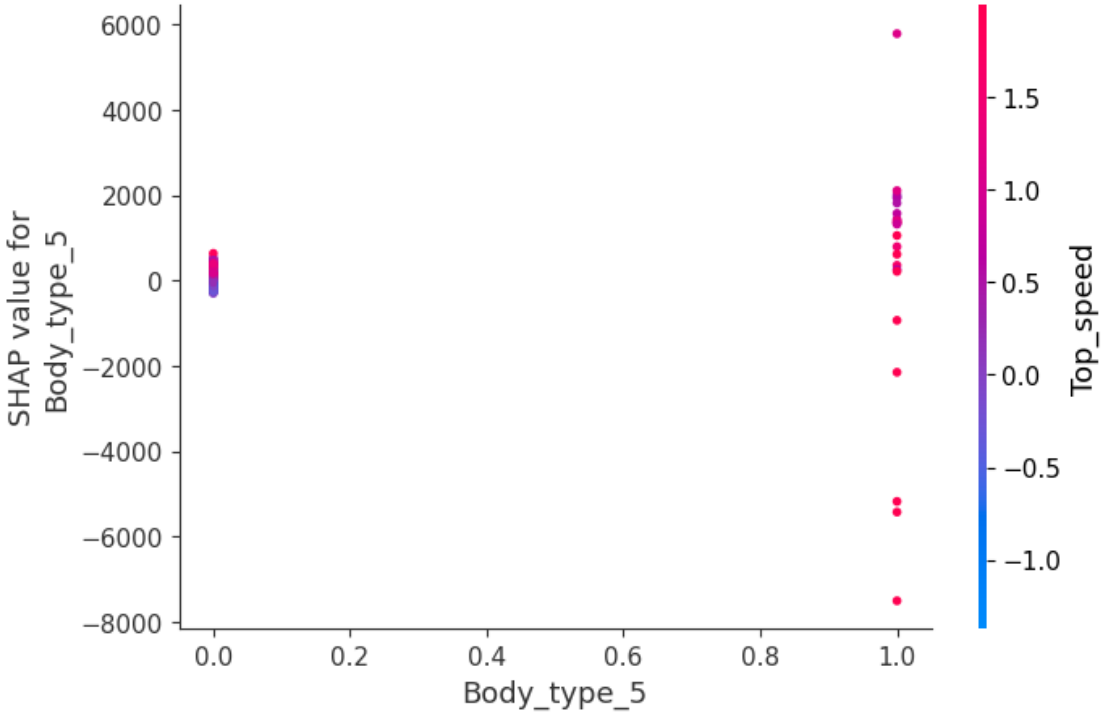
Body_type_1


SHAP value for
Body_type_1
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
Body_type_1
Runned_Miles
1.5
1.0
0.5
0.0
−0.5
−1.0

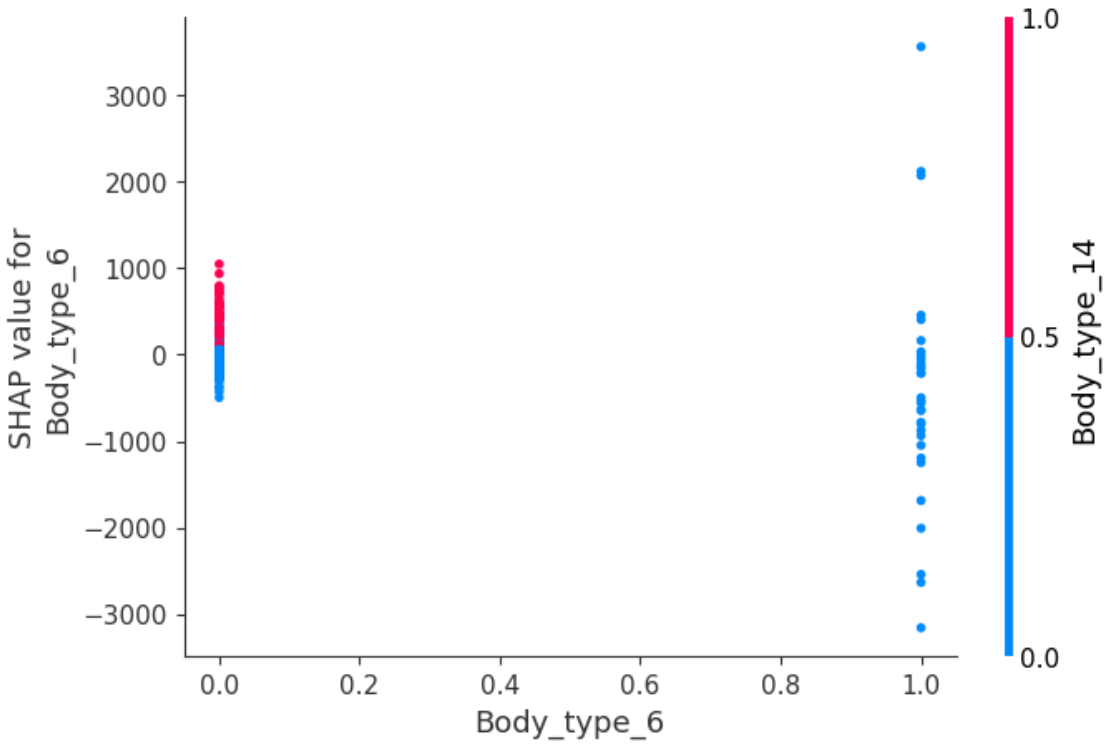
Body_type_4


SHAP value for
Body_type_4
4000
2000
0
−2000
−4000
−6000
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_4
Door_num
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5

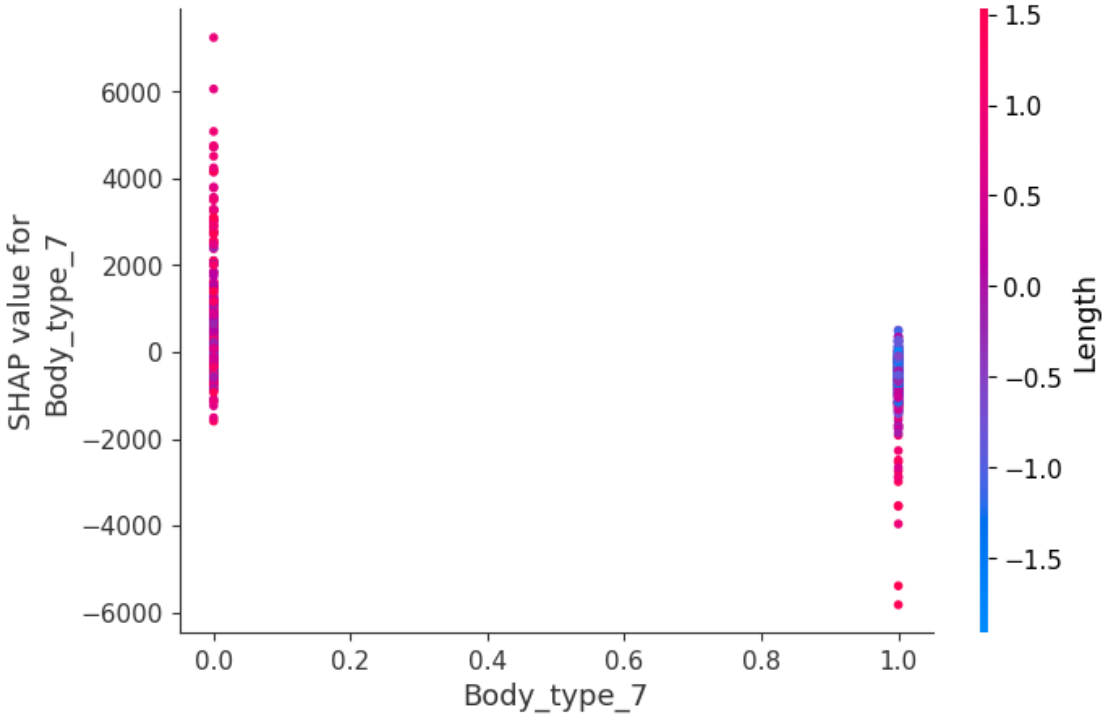
Body_type_5



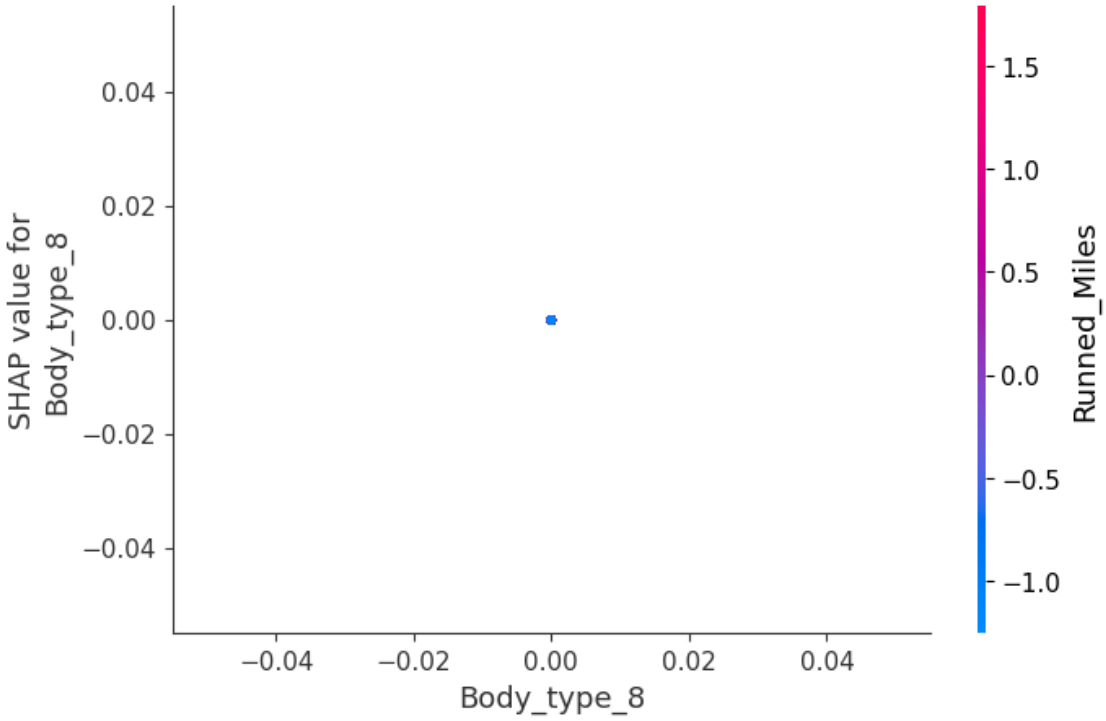
Body_type_6


SHAP value for
Body_type_6
3000
2000
1000
0
−1000
−2000
−3000
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_6
1.0
0.5
0.0
Body_type_14

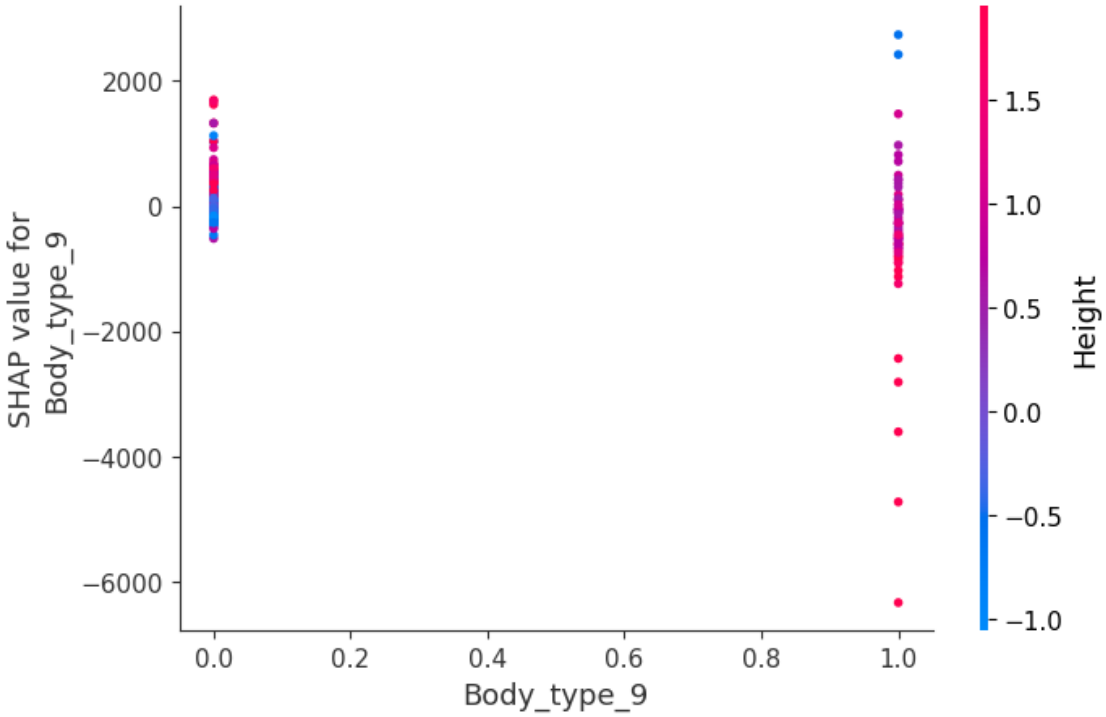
Body_type_7


SHAP value for
Body_type_7
6000
4000
2000
0
−2000
−4000
−6000
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_7
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Length

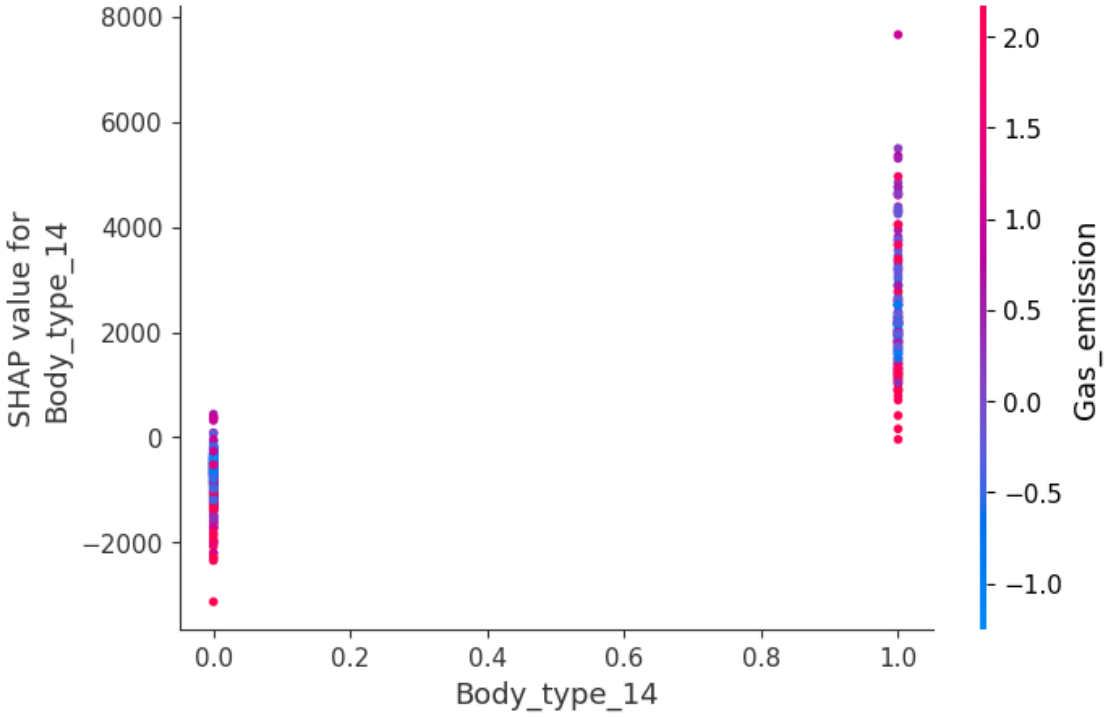
Body_type_8



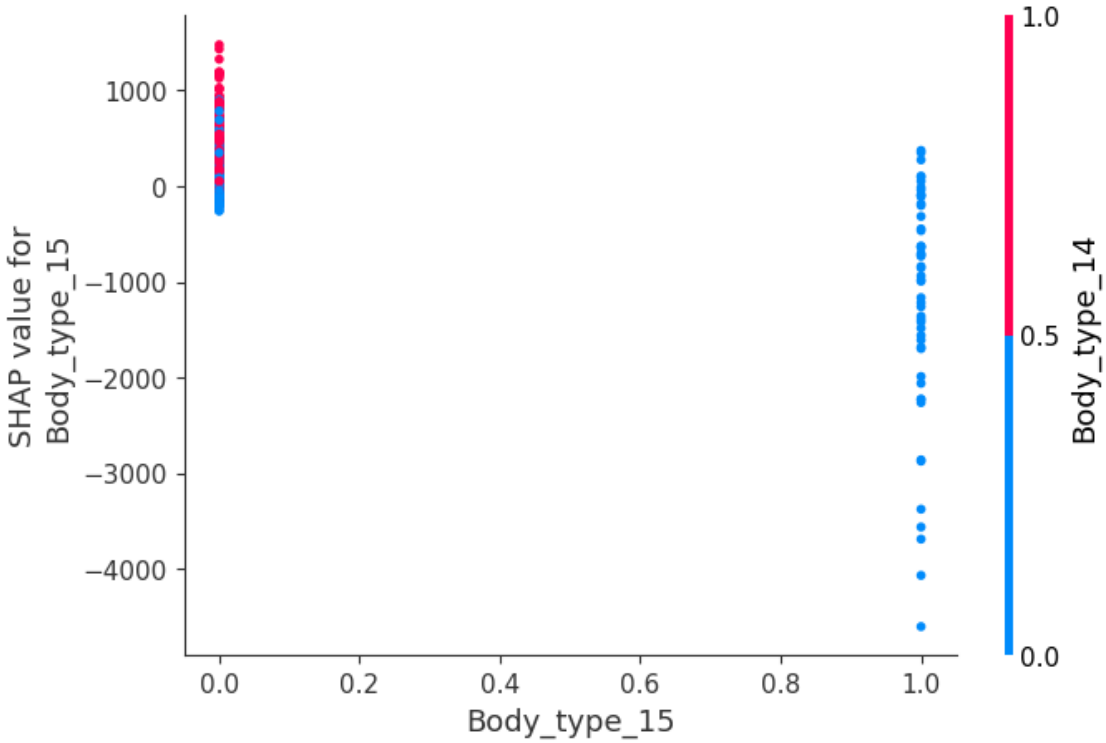
Body_type_9


SHAP value for
Body_type_9
2000
0
−2000
−4000
−6000
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_9
1.5
1.0
0.5
0.0
−0.5
−1.0
Height

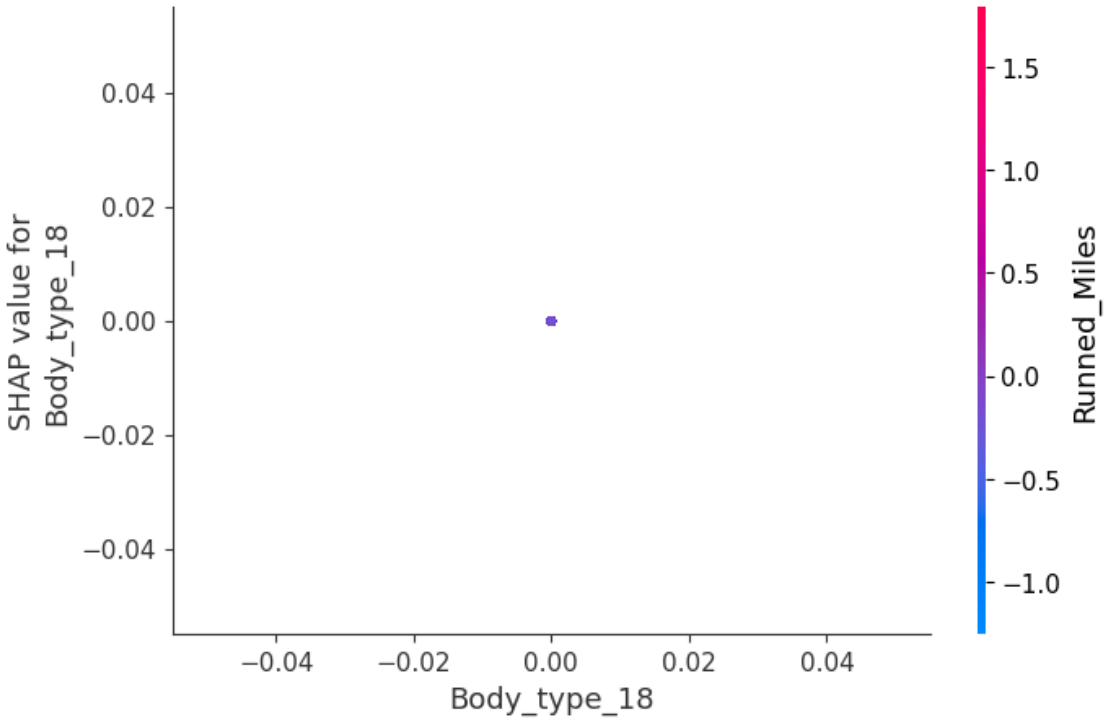
Body_type_14


SHAP value for
Body_type_14
8000
6000
4000
2000
0
−2000
0.0
0.2
0.4
0.6
0.8
1.0
Body_type_14
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
Gas_emission

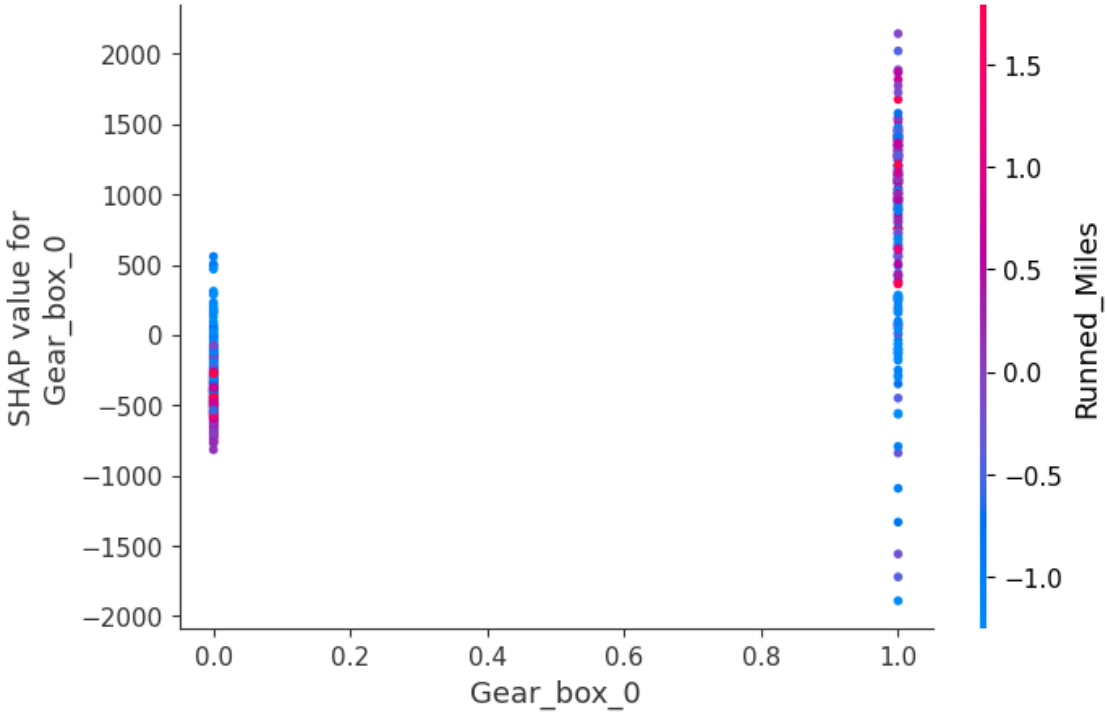
Body_type_15



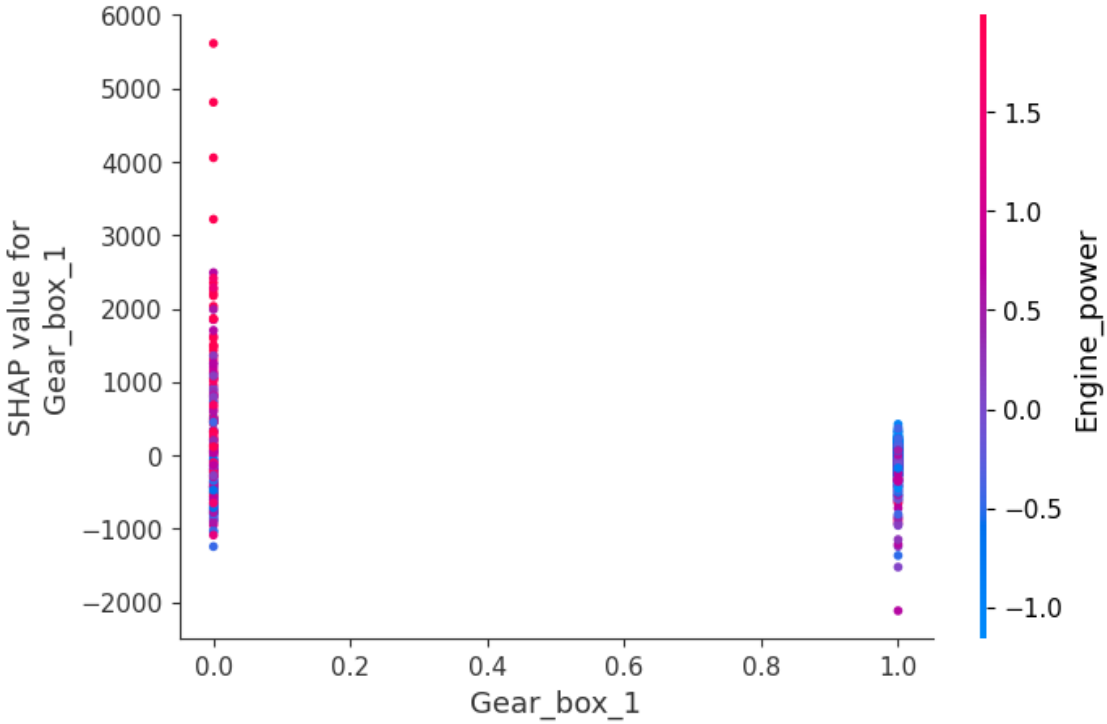
Body_type_18


SHAP value for
Body_type_18
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
Body_type_18
Runned_Miles
1.5
1.0
0.5
0.0
−0.5
−1.0

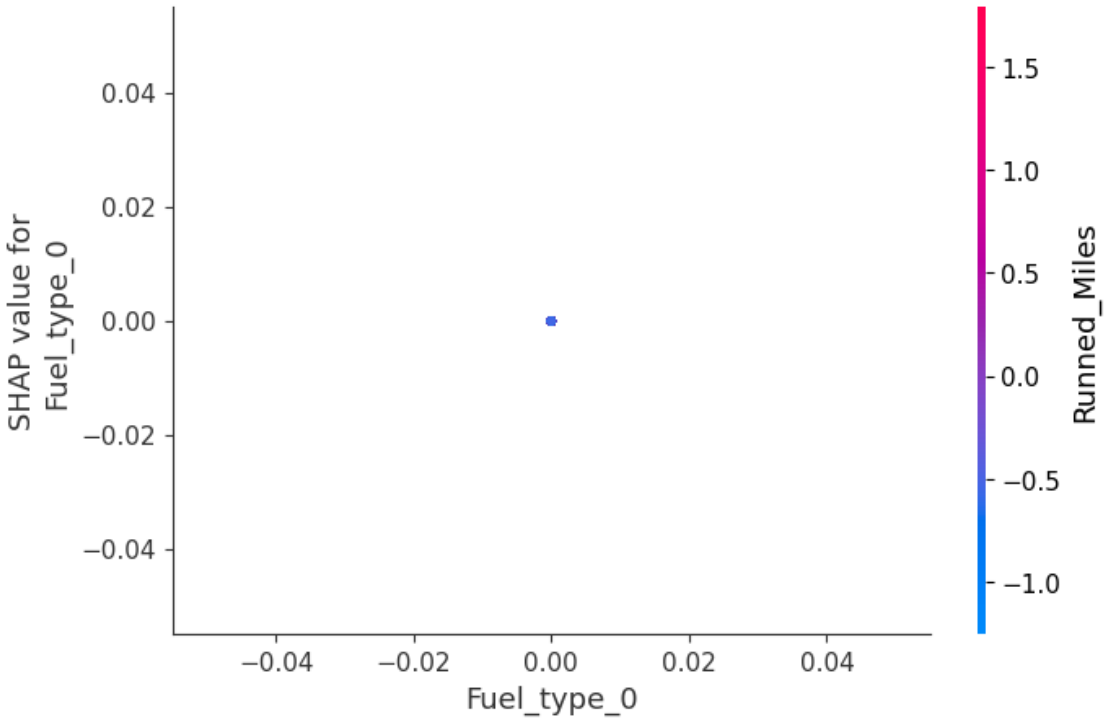
Gear_box_0


SHAP value for
Gear_box_0
2000
1500
1000
500
0
−500
−1000
−1500
−2000
0.0
0.2
0.4
0.6
0.8
1.0
Gear_box_0
Runned_Miles
1.5
1.0
0.5
0.0
−0.5
−1.0

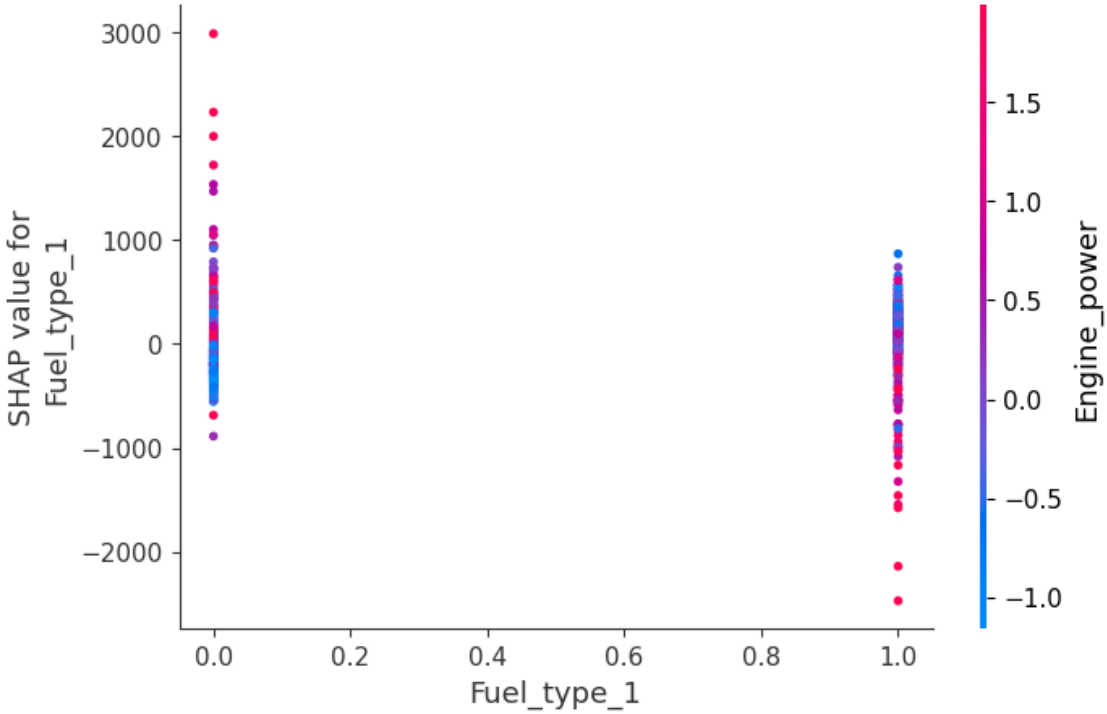
Gear_box_1



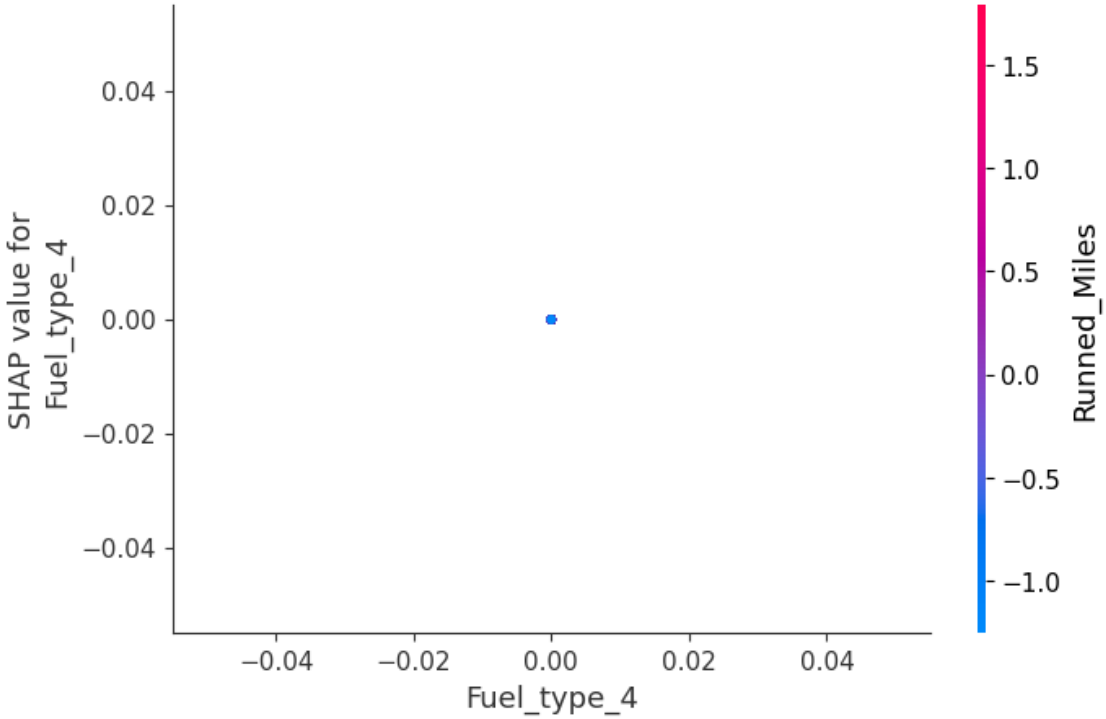
Fuel_type_0


SHAP value for
Fuel_type_0
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
Fuel_type_0
Runned_Miles
1.5
1.0
0.5
0.0
−0.5
−1.0

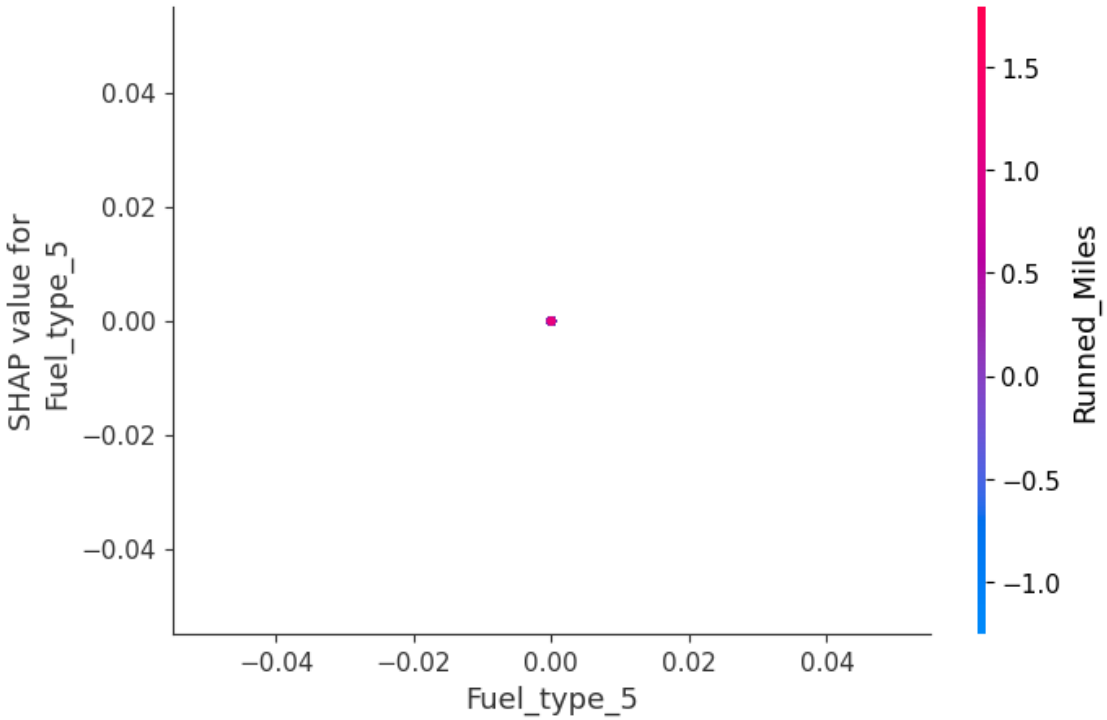
Fuel_type_1


SHAP value for
Fuel_type_1
3000
2000
1000
0
−1000
−2000
0.0
0.2
0.4
0.6
0.8
1.0
Fuel_type_1
Engine_power
1.5
1.0
0.5
0.0
−0.5
−1.0

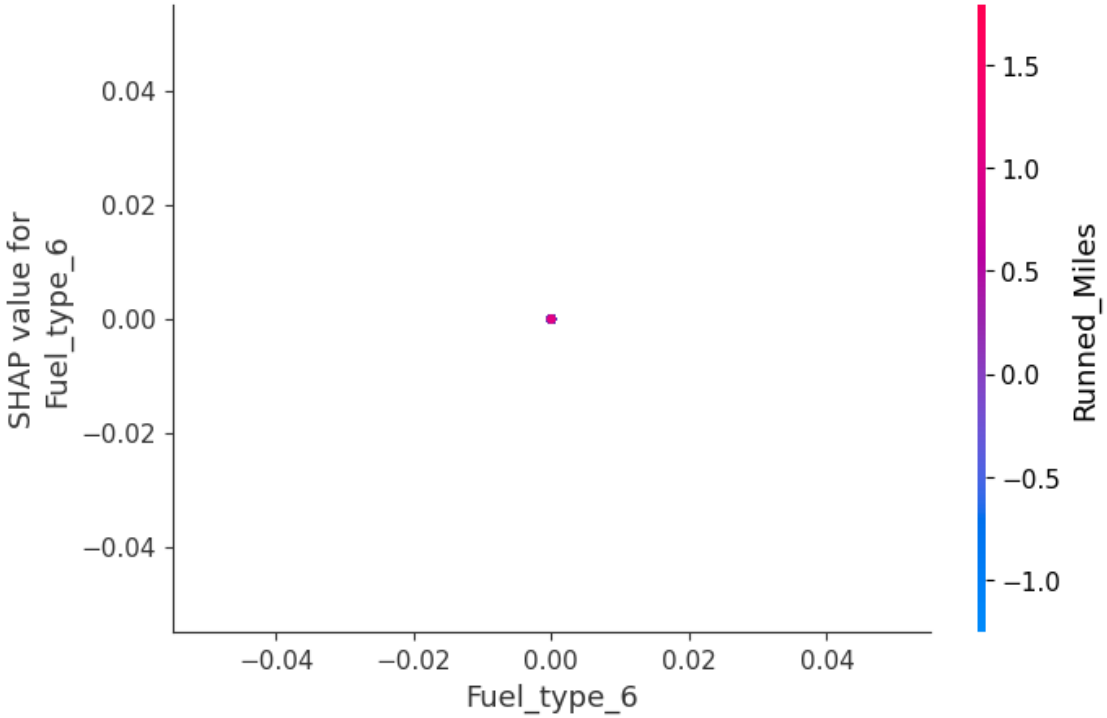
Fuel_type_4



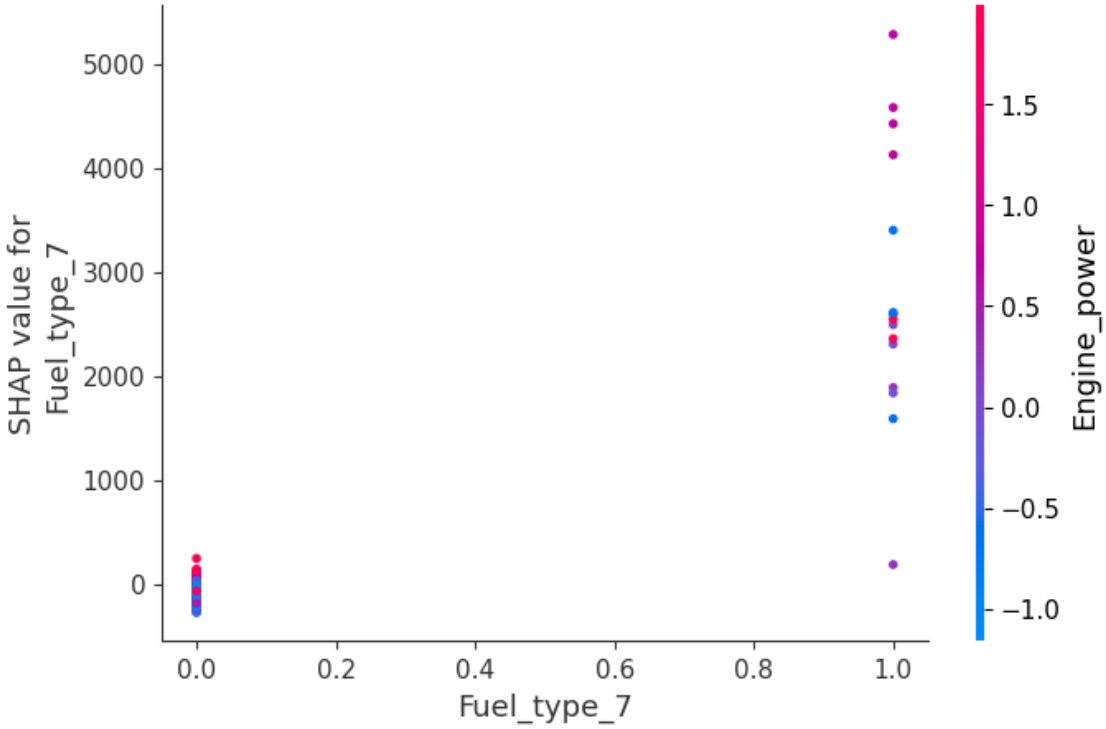
Fuel_type_5


SHAP value for
Fuel_type_5
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
Fuel_type_5
Runned_Miles
1.5
1.0
0.5
0.0
−0.5
−1.0

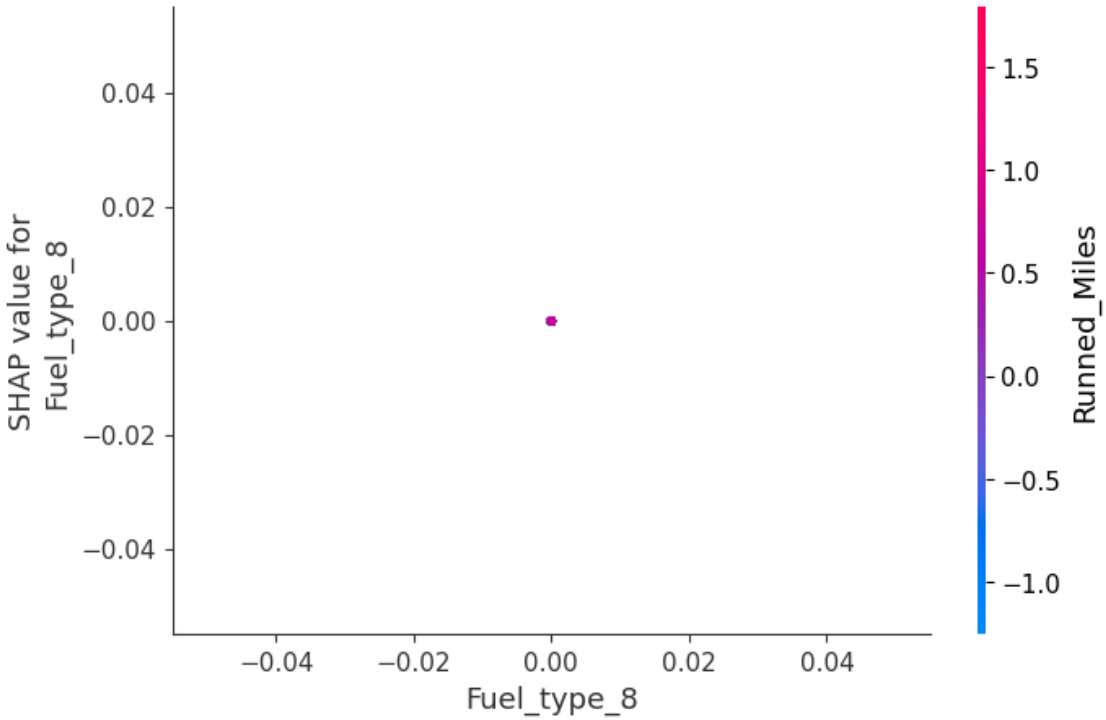
Fuel_type_6


SHAP value for
Fuel_type_6
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
Fuel_type_6
Runned_Miles
1.5
1.0
0.5
0.0
−0.5
−1.0

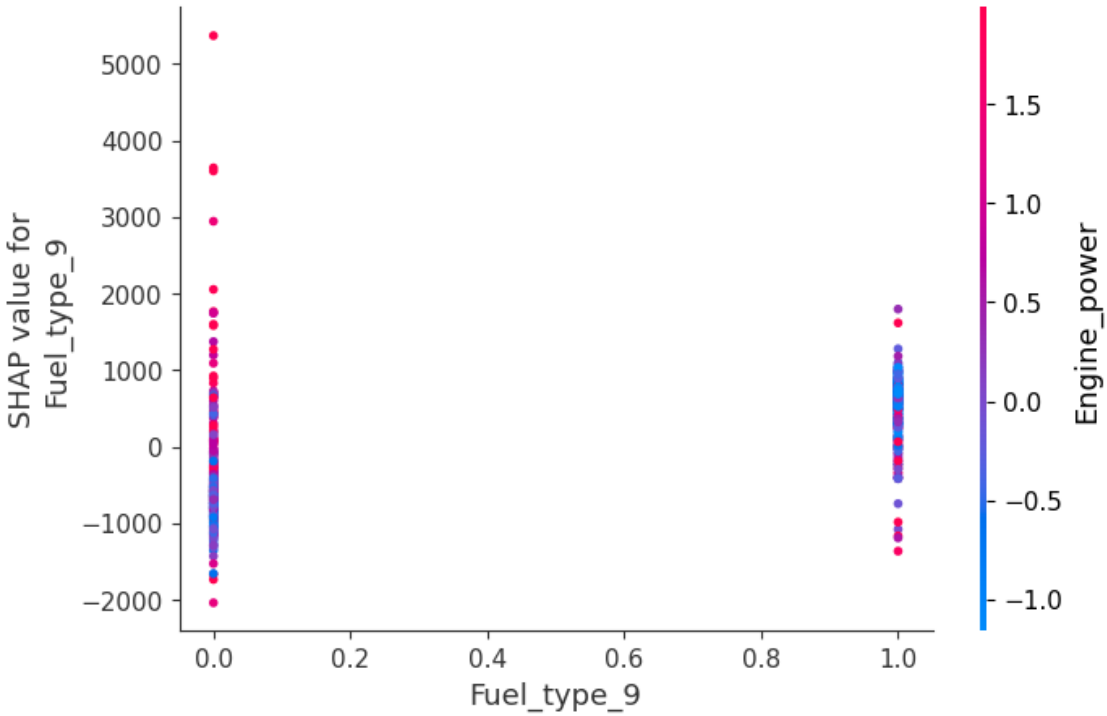
Fuel_type_7



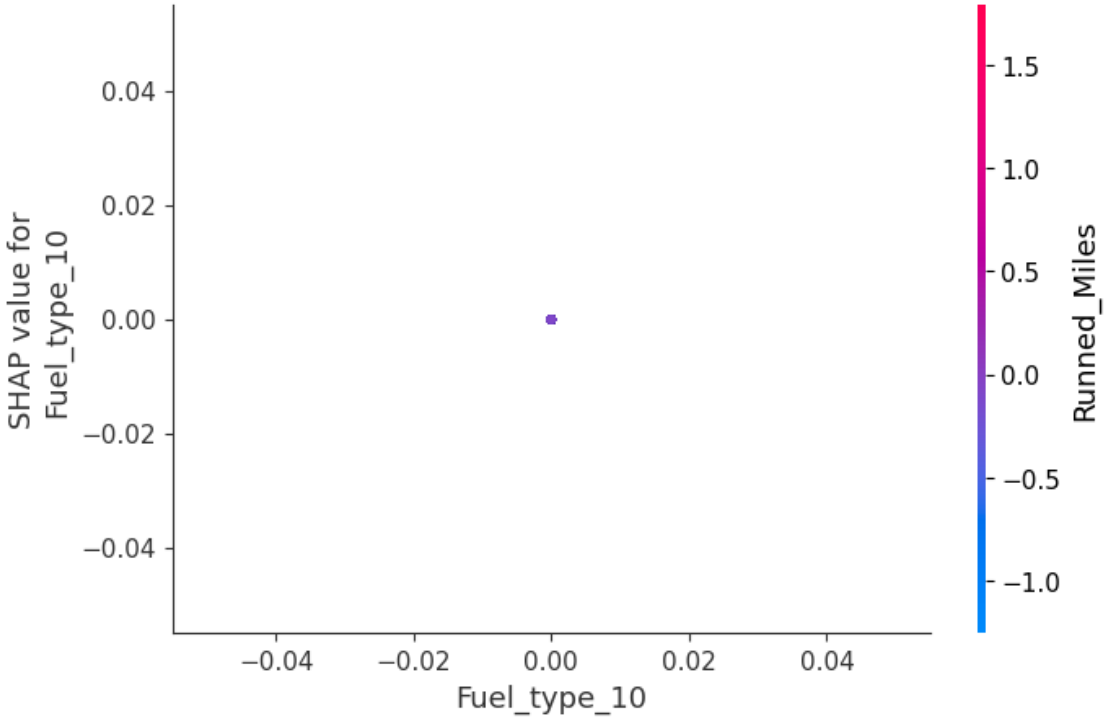
Fuel_type_8


SHAP value for
Fuel_type_8
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04
Fuel_type_8
1.5
1.0
0.5
0.0
−0.5
−1.0
Runned_Miles

Fuel_type_9


SHAP value for
Fuel_type_9
5000
4000
3000
2000
1000
0
−1000
−2000
0.0
0.2
0.4
0.6
0.8
1.0
Fuel_type_9
1.5
1.0
0.5
0.0
−0.5
−1.0
Engine_power

Fuel_type_10



MLP with PCA components

[ ]:
```
y, X = preprocess_Ad_Table_Trim(Ad_table, Trim).final_set(
                standardization=True,
                remove_outliers=True,
                columns_to_drop=columns_to_drop)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=RANDOM_STATE)

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.
 ↪2, random_state=RANDOM_STATE)
```

[ ]:
```
pca = PCA()
X_pca = pca.fit_transform(X_train)
```

[ ]:
```
n_components = 11
pca = PCA(n_components=n_components)
X_train_pca = pca.fit_transform(X_train)
X_val_pca = pca.transform(X_val)
```

```
X_test_pca = pca.transform(X_test)
```

[ ]:
```
X_train_pca_tensor = torch.tensor(X_train_pca, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32).view(-1, 1)

X_val_pca_tensor = torch.tensor(X_val_pca, dtype=torch.float32)
y_val_tensor = torch.tensor(y_val.values, dtype=torch.float32).view(-1, 1)

X_test_pca_tensor = torch.tensor(X_test_pca, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32).view(-1, 1)

train_dataset_pca = TensorDataset(X_train_pca_tensor, y_train_tensor)
eval_dataset_pca = TensorDataset(X_val_pca_tensor, y_val_tensor)
test_dataset_pca = TensorDataset(X_test_pca_tensor, y_test_tensor)
```

[ ]:
```
# Parameters to swap
BATCH_SIZE = 64
learning_rate = 0.08
EPOCHS = 30
activation_function = "Relu"



# Setup loaders
test_loader = DataLoader(test_dataset_pca, batch_size=BATCH_SIZE, shuffle=False)
train_loader = DataLoader(train_dataset_pca, batch_size=BATCH_SIZE,␣
 ↪shuffle=True)
valid_loader = DataLoader(eval_dataset_pca, batch_size=BATCH_SIZE,␣
 ↪shuffle=False)

# fixed parameters

# Number of input features
num_features = n_components
# Total number of layers
num_layers = 3
# Sizes of each layer
layer_sizes = [8, 5, 2]
# Dropout probabilities
Dropout_probas=[0,0,0]

best_valid_loss = float('inf')

# Setup the model
model = MLP_regression(num_features,
                       num_layers,
                       layer_sizes,
```

```
                          activation_function,
                          Dropout_probas)

# Setup criterion and the optimizer
criterion = torch.nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

for epoch in trange(EPOCHS, desc="Epochs"):

    start_time = time.monotonic()

    train_loss = train(model, train_loader, optimizer, criterion)
    valid_loss = evaluate(model, valid_loader, criterion)

    if valid_loss < best_valid_loss:
        best_valid_loss = valid_loss
        torch.save(model, '/content/drive/MyDrive/ML BigProject/model_MLP_pca.
 ↪pt')
        torch.save(model.state_dict(), '/content/drive/MyDrive/ML BigProject/
 ↪model_MLP_weights_pca.pt')
        print("New best model ----")
    end_time = time.monotonic()

    epoch_mins, epoch_secs = epoch_time(start_time, end_time)
    print("valid loss : ", valid_loss, "train loss : ", train_loss, "epoch : ",␣
 ↪epoch)
```

```
Epochs:   0%|          | 0/30 [00:00<?, ?it/s]

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

New best model ----
valid loss :  3217.616678672262 train loss :  3716.5834237221975 epoch :  0

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

New best model ----
valid loss :  3187.2102925819927 train loss :  3218.843530527373 epoch :  1

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

New best model ----
valid loss :  3127.693182191754 train loss :  3154.419179900322 epoch :  2

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
```

```
New best model ----
valid loss :  3092.768036190826 train loss :  3076.469509508304 epoch :  3

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

New best model ----
valid loss :  3020.7580530147743 train loss :  3065.422502643345 epoch :  4

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

New best model ----
valid loss :  3017.9566476349783 train loss :  3049.9212348412734 epoch :  5

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

New best model ----
valid loss :  3016.9157318417388 train loss :  3045.125817550652 epoch :  6

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

valid loss :  3023.8194664681314 train loss :  3046.745476143144 epoch :  7

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

valid loss :  3025.0451756845605 train loss :  3047.3745357216794 epoch :  8

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

New best model ----
valid loss :  2979.575749584236 train loss :  3042.0514413003225 epoch :  9

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

valid loss :  2997.6529028562036 train loss :  3034.1616540448745 epoch :  10

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

valid loss :  3017.548894840656 train loss :  3020.027910207037 epoch :  11

Training:   0%|          | 0/2019 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]

valid loss :  2993.5891350170173 train loss :  3016.1505662635627 epoch :  12

Training:   0%|          | 0/2019 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  3125.2512704981436 train loss :  3012.558039868805 epoch :  13
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  2998.8743437209932 train loss :  3006.305120520098 epoch :  14
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  3022.4571008663365 train loss :  3003.892634958132 epoch :  15
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
New best model ----
valid loss :  2962.3593394666614 train loss :  2995.51913456683 epoch :  16
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
New best model ----
valid loss :  2958.524656994508 train loss :  2991.2545689605968 epoch :  17
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
New best model ----
valid loss :  2954.969814791538 train loss :  2988.322182007743 epoch :  18
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  3142.9590701868037 train loss :  2991.8439087700053 epoch :  19
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  2957.06441033996 train loss :  2991.071104838507 epoch :  20
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  2959.7008996944614 train loss :  2990.1306809552416 epoch :  21
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  3003.204582592048 train loss :  2987.798474731748 epoch :  22
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  2964.2208264039295 train loss :  2985.0511107612447 epoch :  23
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  2963.337728186881 train loss :  2983.2606005278953 epoch :  24
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
New best model ----
valid loss :  2937.932219794245 train loss :  2985.1246329426117 epoch :  25
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  2938.9517902034345 train loss :  2975.344725413745 epoch :  26
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
New best model ----
valid loss :  2930.949339369972 train loss :  2983.1572650155554 epoch :  27
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  3035.699905486154 train loss :  2982.1948060200557 epoch :  28
Training:   0%|          | 0/2019 [00:00<?, ?it/s]
Evaluating:   0%|          | 0/505 [00:00<?, ?it/s]
valid loss :  2952.9063665106746 train loss :  2981.067168841804 epoch :  29
```

[ ]:
```python
# Test the model with PCA
best_model = torch.load('/content/drive/MyDrive/ML BigProject/model_MLP_pca.pt')
test_loss = evaluate(best_model, test_loader, criterion)
print(f"Test Loss: {test_loss:.3f}")
```

```
Evaluating:   0%|          | 0/631 [00:00<?, ?it/s]
Test Loss: 3056.089
```

6.2 Images data

[ ]:
```python
# Connexion to a device
random.seed(RANDOM_STATE)
torch.manual_seed(RANDOM_STATE)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

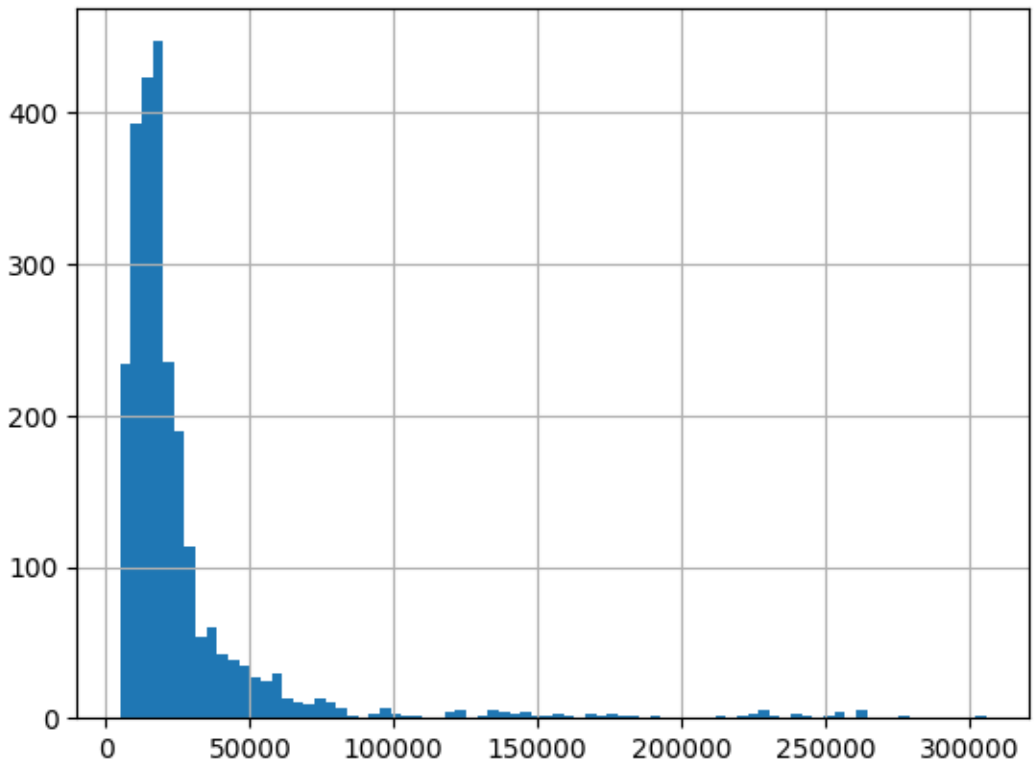
6.2.1 Data

[ ]:
```
# unpack all images
import shutil
shutil.unpack_archive(filename ='/content/drive/MyDrive/ML BigProject/data/
 ↪Confirmed_fronts.zip', extract_dir = 'confirmed_fronts')
```

[ ]:
```
autosWithNames = pd.read_csv("/content/drive/MyDrive/ML BigProject/data/
 ↪autosWithNames.csv", sep=",")

img_dir='confirmed_fronts/'

# Normalize columns names
autosWithNames.rename(columns={'brand': 'Maker',
                               'model_name': 'Genmodel',
                               'model_id': 'Genmodel_ID',
                               'model_year' : 'Year'}, inplace=True)
autosWithNames.drop(columns = 'bodytype', inplace = True)

# Merge price table with the AutoWitgNames table (to make the link with images␣
 ↪and give them a price)
df = pd.merge(Price_table, autosWithNames, on=['Maker', 'Genmodel',␣
 ↪'Genmodel_ID', 'Year'], how='inner')
```

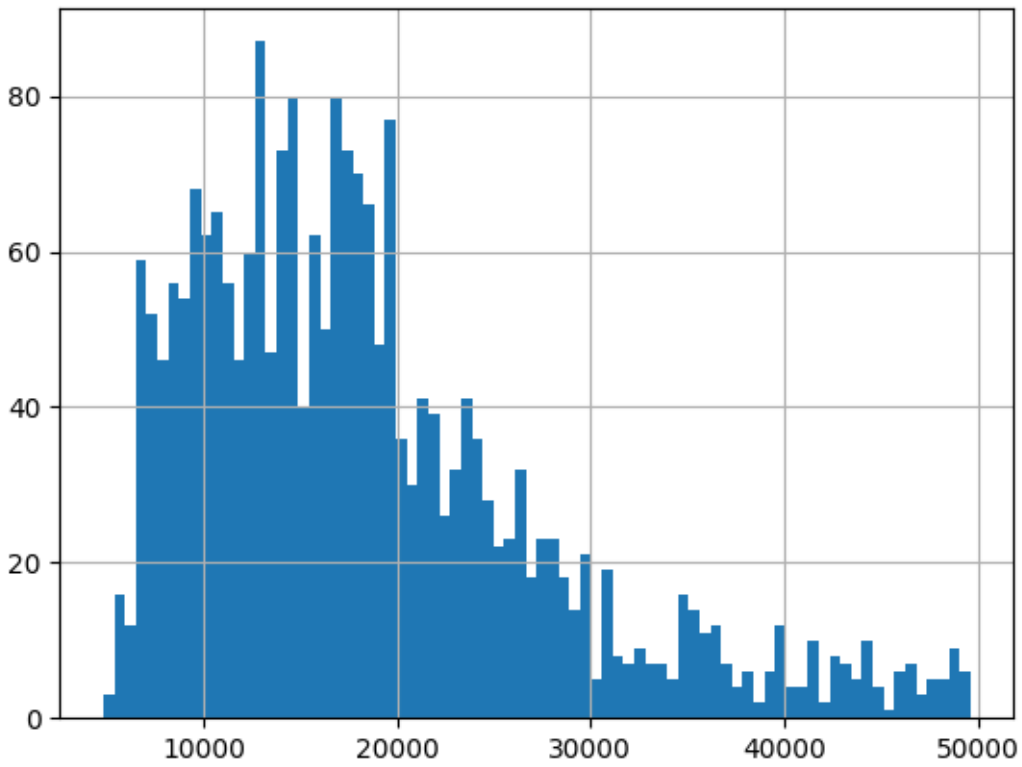
[ ]:
```
df['Entry_price'].hist(bins=80)
```

[ ]:
```
<Axes: >
```



[ ]:
```
df = df[df['Entry_price'] <= 50000]
df['Entry_price'].hist(bins=80)
```

[ ]: <Axes: >



[ ]:
```
nb_train = int(len(df) * 0.7)
nb_valid = int(len(df) * 0.1)
nb_test = int(len(df) * 0.2)


train_data = df[:nb_train] # split the datas into train valid and test
test_data = df[:nb_test]
valid_data = df[:nb_valid]

# Those arent the actual number of images, but the number of different model␣
 ↪per dataset. The each model can have different images
len(train_data), len(test_data), len(valid_data)
```

[ ]:
```
(1685, 481, 240)
```

[ ]:
```
# Parameters of images transformation
# those are pretrained mean and stds of the ImageNet dataset
pretrained_means = [0.485, 0.456, 0.406]
pretrained_stds = [0.229, 0.224, 0.225]


# Train images transformations
```

```
train_transforms = transforms.Compose([
                            transforms.Resize(size=(224,224)),
                            transforms.RandomRotation(degrees=45),
                            transforms.RandomHorizontalFlip(p=0.2),
                            transforms.RandomVerticalFlip(p=0.2),
                            transforms.ToTensor(),
                            transforms.Normalize(mean=pretrained_means,␣
 ↪std=pretrained_stds)
                                      ])

# Test images transformations

test_transforms = transforms.Compose([
                           transforms.Resize(size=(224,224)),
                           transforms.ToTensor(),
                           transforms.Normalize(mean=pretrained_means,␣
 ↪std=pretrained_stds),
                                     ])


def get_splitted_data(BATCH_SIZE,
                      train_data,
                      valid_data,
                      test_data,
                      train_transforms,
                      test_transforms):


    # ----------------------------------------------------------------

    # plug train_data in the preprocessing class to get the train_dataset
    train_dataset = DVMdataset(data=train_data,img_dir='confirmed_fronts/
 ↪',transforms=train_transforms)

    # plug the train_dataset in a DataLoader (then ready to get trained)
    train_loader = DataLoader(dataset=train_dataset,
                              batch_size=BATCH_SIZE,
                              shuffle=True,
                              drop_last=True)

    # ----------------------------------------------------------------

    # plug valid_data in the preprocessing class to get the valid_dataset
    valid_dataset = DVMdataset(data=valid_data,img_dir='confirmed_fronts/
 ↪',transforms=test_transforms)
    # Valid dataset is transformed as the test set
```

```
    # plug the valid_dataset in a DataLoader
    valid_loader = DataLoader(dataset=valid_dataset,
                              batch_size=BATCH_SIZE,
                              shuffle=False)

    # ----------------------------------------------------------------

    # plug test_data in the preprocessing class to get the test_dataset
    test_dataset = DVMdataset(data=test_data,img_dir='confirmed_fronts/
 ↪',transforms=test_transforms)

    # plug the test_dataset in a DataLoader
    test_loader = DataLoader(dataset=test_dataset,
                             batch_size=BATCH_SIZE,
                             shuffle=False)

    return train_loader, valid_loader, test_loader, train_dataset,␣
 ↪valid_dataset, test_dataset
```

[ ]:
```
train_loader, valid_loader, test_loader, train_dataset, valid_dataset,␣
 ↪test_dataset = get_splitted_data(100,
                                                                    ␣
 ↪                          train_data,
                                                                    ␣
 ↪                          valid_data,
                                                                    ␣
 ↪                          test_data,
                                                                    ␣
 ↪                          None,
                                                                    ␣
 ↪                          None)


# Print number of images labelled for each dataset
print('Train dataset lenght : ', train_dataset.__len__(), "\n",
      'Valid dataset lenght : ', valid_dataset.__len__(),"\n",
    'Test dataset lenght : ', test_dataset.__len__())
```

```
Train dataset lenght :  45832
 Valid dataset lenght :  13125
 Test dataset lenght :  18480
```

[ ]:
```
liste_images = []
liste_prices = []
for i in range(5):
  num = random.randint(1,test_dataset.__len__())
```

```
  image = test_dataset.__getitem__(num)
  liste_images.append(image[0])
  liste_prices.append(image[1])

fig, axes = plt.subplots(1, len(liste_images), figsize=(15, 5))

for ax, img, price in zip(axes, liste_images, liste_prices):
    ax.imshow(img)
    ax.axis('off')
    ax.text(0.5, -0.1, f"${price:.2f}", fontsize=12, ha='center', transform=ax.
 ↪transAxes)
```

$24435.00 $36527.00 $18810.00 $26095.00 $5810.00

[ ]:
```
train_loader, valid_loader, test_loader, train_dataset, valid_dataset,␣
 ↪test_dataset = get_splitted_data(100,
                                                                        ␣
 ↪                          train_data,
                                                                        ␣
 ↪                          valid_data,
                                                                        ␣
 ↪                          test_data,
                                                                        ␣
 ↪                          train_transforms,
                                                                        ␣
 ↪                          test_transforms)
```

[ ]:
```
liste_images = []
liste_prices = []
for i in range(5):
  num = random.randint(1,train_dataset.__len__())
  image = train_dataset.__getitem__(num)
  liste_images.append(image[0])
  liste_prices.append(image[1])

fig, axes = plt.subplots(1, len(liste_images), figsize=(15, 5))

for ax, img, price in zip(axes, liste_images, liste_prices):
    #ax.imshow(img)
```

```
    ax.imshow(img.permute(1, 2, 0))
    ax.axis('off')
    ax.text(0.5, -0.1, f"${price:.2f}", fontsize=12, ha='center', transform=ax.
 ↪transAxes)
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```

[ ]:
```
liste_images = []
liste_prices = []
for i in range(5):
  num = random.randint(1,test_dataset.__len__())
  image = test_dataset.__getitem__(num)
  liste_images.append(image[0])
  liste_prices.append(image[1])

fig, axes = plt.subplots(1, len(liste_images), figsize=(15, 5))

for ax, img, price in zip(axes, liste_images, liste_prices):
    #ax.imshow(img)
    ax.imshow(img.permute(1, 2, 0))
    ax.axis('off')
    ax.text(0.5, -0.1, f"${price:.2f}", fontsize=12, ha='center', transform=ax.
 ↪transAxes)
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
```

```
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```

6.2.2 Resnet18 pre-trained model

[ ]:
```
 # import resnet18 pretrained model
pretrained_model = models.resnet18(pretrained=True)

# check the last layer input size
number_inputs_last_layer = pretrained_model.fc.in_features
print("Original last layer size : ", pretrained_model.fc)

# change the output size of the last layer, but keep the same number of input
pretrained_model.fc = nn.Linear(number_inputs_last_layer, 1)
print("Transformed last layer size : ", pretrained_model.fc)
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be
removed in the future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are
deprecated since 0.13 and may be removed in the future. The current behavior is
equivalent to passing `weights=ResNet18_Weights.IMAGENET1K_V1`. You can also use
`weights=ResNet18_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to
/root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth
100%|      | 44.7M/44.7M [00:00<00:00, 184MB/s]

Original last layer size :  Linear(in_features=512, out_features=1000,
bias=True)
Transformed last layer size :  Linear(in_features=512, out_features=1,
bias=True)
```

```
[ ]: pretrained_model
```

```
[ ]: ResNet(
       (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
     bias=False)
       (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
     track_running_stats=True)
       (relu): ReLU(inplace=True)
       (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
     ceil_mode=False)
       (layer1): Sequential(
         (0): BasicBlock(
           (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
     bias=False)
           (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
     track_running_stats=True)
           (relu): ReLU(inplace=True)
           (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
     bias=False)
           (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
     track_running_stats=True)
         )
         (1): BasicBlock(
           (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
     bias=False)
           (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
     track_running_stats=True)
           (relu): ReLU(inplace=True)
           (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
     bias=False)
           (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
     track_running_stats=True)
         )
       )
       (layer2): Sequential(
         (0): BasicBlock(
           (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
     1), bias=False)
           (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
     track_running_stats=True)
           (relu): ReLU(inplace=True)
           (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
     1), bias=False)
           (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
     track_running_stats=True)
           (downsample): Sequential(
             (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
```

```
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
```

```
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=1, bias=True)
)
```

[ ]:
```python
def train(model, iterator, optimizer, criterion, device):
    epoch_loss = 0
    model.train()
    for (x, y) in tqdm(iterator, desc="Training", leave=False):
        x = x.to(device)
        y = y.float()
        y = y.to(device)
        y = y.squeeze()

        optimizer.zero_grad()

        y_pred = model(x)

        mse = criterion(y_pred, y)
        loss = torch.sqrt(mse)
```

```
        loss.backward()
        optimizer.step()

        epoch_loss += loss.item()
    return epoch_loss / len(iterator)

def evaluate(model, iterator, criterion, device):
    epoch_loss = 0
    model.eval()
    with torch.no_grad():
        for (x, y) in tqdm(iterator, desc="Evaluating", leave=False):
            x = x.to(device)

            y = y.float()
            y = y.to(device)
            y = y.squeeze()

            y_pred = model(x)


            mse = criterion(y_pred, y)
            loss = torch.sqrt(mse)

            epoch_loss += loss.item()
    return epoch_loss / len(iterator)

def epoch_time(start_time, end_time):
    elapsed_time = end_time - start_time
    elapsed_mins = int(elapsed_time / 60)
    elapsed_secs = int(elapsed_time - (elapsed_mins * 60))
    return elapsed_mins, elapsed_secs
```

6.2.3 Hyperparameters optimization

[ ]:
```
%%capture
!pip install wandb --upgrade
import wandb
wandb.login()
```

[ ]:
```
sweep_config = {
    'method': 'random'
    }

metric = {
  'name': 'valid loss',
  'goal': 'minimize'
}
```

```
sweep_config['metric'] = metric

parameters_dict = {

    'learning_rate': {
        'distribution' : 'uniform',
        'min' : 5e-4,
        'max' : 0.07,
    },
    'batch_size' : {
        'distribution': 'q_log_uniform_values',
        'q': 8,
        'min': 30,
        'max': 200,
    },
}

sweep_config['parameters'] = parameters_dict
```

[ ]:
```
sweep_id = wandb.sweep(sweep_config, project="Car_prices_prediction")
```

```
Create sweep with ID: h4p30t8g
Sweep URL: https://wandb.ai/ojlt/Car_prices_prediction/sweeps/h4p30t8g
```

[ ]:
```
def wandb_train(config=None):
  with wandb.init(config=config):

    config=wandb.config

    learning_rate = config.learning_rate
    BATCH_SIZE = config.batch_size

    train_loader, valid_loader, test_loader, train_dataset, valid_dataset,␣
 ↪test_dataset = get_splitted_data(BATCH_SIZE,
                                                                          ␣
 ↪                              train_data,
                                                                          ␣
 ↪                              valid_data,
                                                                          ␣
 ↪                              test_data,
                                                                          ␣
 ↪                              train_transforms,
                                                                          ␣
 ↪                              test_transforms)
    # import resnet50 pretrained model
    pretrained_model = models.resnet18(pretrained=True)
```

```
    # check the last layer input size
    number_inputs_last_layer = pretrained_model.fc.in_features

    # change the output size of the last layer, but keep the same number of␣
↪input
    pretrained_model.fc = nn.Linear(number_inputs_last_layer, 1)
    pretrained_model.fc



    optimizer = optim.Adam(pretrained_model.parameters(), lr=learning_rate)
    criterion = nn.MSELoss()
    model = pretrained_model.to(device)
    criterion = criterion.to(device)

    EPOCHS = 2
    for epoch in trange(EPOCHS, desc="Epochs"):

        start_time = time.monotonic()

        train_loss = train(model, train_loader, optimizer, criterion, device)
        valid_loss = evaluate(model, valid_loader, criterion, device)


        end_time = time.monotonic()

        epoch_mins, epoch_secs = epoch_time(start_time, end_time)

        wandb.log({"valid loss": valid_loss, "train loss": train_loss, "epoch":␣
↪epoch})
```

[ ]:
```
start_time = time.time()
wandb.agent("h4p30t8g", wandb_train, count=40)
elapsed = (time.time() - start_time)/60
print(f'Total Time: {elapsed:.2f} min')
```

```
wandb: Agent Starting Run: npx019h9 with config:
wandb:     batch_size: 136
wandb:     learning_rate: 0.034141272877410436
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be
removed in the future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are
deprecated since 0.13 and may be removed in the future. The current behavior is
equivalent to passing `weights=ResNet18_Weights.IMAGENET1K_V1`. You can also use
`weights=ResNet18_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/337 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([136])) that is different to the
input size (torch.Size([136, 1])). This will likely lead to incorrect results
due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/97 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([69])) that is different to the
input size (torch.Size([69, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Training:   0%|          | 0/337 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/97 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: co4iax8o with config:
wandb:     batch_size: 32
wandb:     learning_rate: 0.0498644771849091

VBox(children=(Label(value='Waiting for wandb.init()…\r'), FloatProgress(value=0.
 ↪011112892100000712, max=1.0…

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([32])) that is different to the
input size (torch.Size([32, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([5])) that is different to the
input size (torch.Size([5, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: vdqp1vqx with config:
wandb:     batch_size: 48
wandb:     learning_rate: 0.015655314670906802

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/954 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([48])) that is different to the
```

```
input size (torch.Size([48, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/274 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([21])) that is different to the
input size (torch.Size([21, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Training:   0%|          | 0/954 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/274 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: rlj1yvrj with config:
wandb:     batch_size: 56
wandb:     learning_rate: 0.044860331793404346

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([56])) that is different to the
input size (torch.Size([56, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: vv9dg964 with config:
wandb:     batch_size: 112
wandb:     learning_rate: 0.05910819269917912

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/409 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([112])) that is different to the
input size (torch.Size([112, 1])). This will likely lead to incorrect results
due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/118 [00:00<?, ?it/s]

Training:   0%|          | 0/409 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/118 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: vw70j59m with config:
wandb:     batch_size: 184
wandb:     learning_rate: 0.0010409668969956238

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/249 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([184])) that is different to the
input size (torch.Size([184, 1])). This will likely lead to incorrect results
due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/72 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([61])) that is different to the
input size (torch.Size([61, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Training:   0%|          | 0/249 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/72 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: ivcw48sc with config:
wandb:     batch_size: 96
wandb:     learning_rate: 0.034033067684774446

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/477 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([96])) that is different to the
input size (torch.Size([96, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
```

```
Evaluating:   0%|          | 0/137 [00:00<?, ?it/s]

Training:   0%|          | 0/477 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/137 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: 69yl9xd1 with config:
wandb:     batch_size: 56
wandb:     learning_rate: 0.04247118581055136

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: jmpuek46 with config:
wandb:     batch_size: 112
wandb:     learning_rate: 0.03046993067365004

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/409 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/118 [00:00<?, ?it/s]

Training:   0%|          | 0/409 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/118 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: cyi13itl with config:
wandb:     batch_size: 80
wandb:     learning_rate: 0.029793271322214505

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/572 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([80])) that is different to the
input size (torch.Size([80, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/165 [00:00<?, ?it/s]

Training:   0%|          | 0/572 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/165 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: 1njhuxbx with config:
wandb:     batch_size: 40
wandb:     learning_rate: 0.03895080858111096

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([40])) that is different to the
input size (torch.Size([40, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: yy5jux1g with config:
wandb:     batch_size: 56
wandb:     learning_rate: 0.06937029062067844

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: o3qqpwuw with config:
wandb:     batch_size: 56
wandb:     learning_rate: 0.035525342168852704

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: pel0kn50 with config:
wandb:     batch_size: 48
wandb:     learning_rate: 0.02962527709578819

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/954 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/274 [00:00<?, ?it/s]

Training:   0%|          | 0/954 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/274 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: g6qjjc4q with config:
wandb:     batch_size: 56
wandb:     learning_rate: 0.038135572309629766

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
wandb: Agent Starting Run: 7qi7wmaw with config:
wandb:     batch_size: 200
wandb:     learning_rate: 0.010517531332882349

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/229 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([200])) that is different to the
input size (torch.Size([200, 1])). This will likely lead to incorrect results
due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/66 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([125])) that is different to the
input size (torch.Size([125, 1])). This will likely lead to incorrect results
due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Training:   0%|          | 0/229 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/66 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.003 MB uploaded\r'),␣
 ↪FloatProgress(value=0.6749090909090909, max=1.0…

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: qfnnabaa with config:
wandb:     batch_size: 40
wandb:     learning_rate: 0.032645772820777526

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: xgxqymtm with config:
wandb:     batch_size: 56
wandb:     learning_rate: 0.012806721331794913

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Sweep Agent: Waiting for job.
wandb: Job received.
wandb: Agent Starting Run: 2nk4pyfj with config:
```

```
wandb:     batch_size: 80
wandb:     learning_rate: 0.030249105802251455
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Epochs:   0%|          | 0/2 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/572 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/165 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/572 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/165 [00:00<?, ?it/s]
```

```
VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
wandb: Agent Starting Run: bifraapa with config:
wandb:     batch_size: 72
wandb:     learning_rate: 0.039766800959749465
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Epochs:   0%|          | 0/2 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/636 [00:00<?, ?it/s]
```

```
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([72])) that is different to the
input size (torch.Size([72, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
```

```
Evaluating:   0%|          | 0/183 [00:00<?, ?it/s]

Training:   0%|          | 0/636 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/183 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.003 MB uploaded\r'),␣
 ↪FloatProgress(value=0.6001294079585895, max=1.0…

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: 5rnocdhi with config:
wandb:     batch_size: 72
wandb:     learning_rate: 0.017437537966628013

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/636 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/183 [00:00<?, ?it/s]

Training:   0%|          | 0/636 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/183 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: 2nvxtsrh with config:
wandb:     batch_size: 40
wandb:     learning_rate: 0.014917817996658237

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: 46p8z5ko with config:
wandb:     batch_size: 40
wandb:     learning_rate: 0.018614424012337944

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: kaz7v81l with config:
wandb:     batch_size: 144
wandb:     learning_rate: 0.011269563421107022
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/318 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([144])) that is different to the
input size (torch.Size([144, 1])). This will likely lead to incorrect results
due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/92 [00:00<?, ?it/s]

Training:   0%|          | 0/318 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/92 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: vtve24tn with config:
wandb:     batch_size: 136
wandb:     learning_rate: 0.012903544096526588

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/337 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/97 [00:00<?, ?it/s]

Training:   0%|          | 0/337 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/97 [00:00<?, ?it/s]
```

```
VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
wandb: Agent Starting Run: ss43q0zz with config:
wandb:     batch_size: 136
wandb:     learning_rate: 0.02664767516370397
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Epochs:   0%|          | 0/2 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/337 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/97 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/337 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/97 [00:00<?, ?it/s]
```

```
VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
wandb: Agent Starting Run: 0h5i0yw2 with config:
wandb:     batch_size: 80
wandb:     learning_rate: 0.06655651260841784
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/572 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/165 [00:00<?, ?it/s]

Training:   0%|          | 0/572 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/165 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: nkwehpmm with config:
wandb:     batch_size: 40
wandb:     learning_rate: 0.001199818400094734

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: j6h5a5ks with config:
wandb:     batch_size: 72
wandb:     learning_rate: 0.06188268088159614

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/636 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/183 [00:00<?, ?it/s]

Training:   0%|          | 0/636 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/183 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: w3jhxfng with config:
wandb:     batch_size: 32
wandb:     learning_rate: 0.03635376144299224

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
wandb: Agent Starting Run: ciwqj30b with config:
wandb:     batch_size: 160
wandb:     learning_rate: 0.01838968187302913

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/286 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([160])) that is different to the
input size (torch.Size([160, 1])). This will likely lead to incorrect results
due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/83 [00:00<?, ?it/s]

Training:   0%|          | 0/286 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/83 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: mziajzik with config:
wandb:     batch_size: 80
wandb:     learning_rate: 0.0026987805079646595

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/572 [00:00<?, ?it/s]
```

```
Evaluating:   0%|          | 0/165 [00:00<?, ?it/s]

Training:   0%|          | 0/572 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/165 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: 670zwy03 with config:
wandb:     batch_size: 104
wandb:     learning_rate: 0.06878701035094043

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/440 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([104])) that is different to the
input size (torch.Size([104, 1])). This will likely lead to incorrect results
due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

Training:   0%|          | 0/440 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/127 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: hf808tn4 with config:
wandb:     batch_size: 48
wandb:     learning_rate: 0.002930115613047949
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/954 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/274 [00:00<?, ?it/s]

Training:   0%|          | 0/954 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/274 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: k12wrquh with config:
wandb:     batch_size: 32
wandb:     learning_rate: 0.015028339493411645

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=0.7241513850955911, max=1.0…

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: dnarv2ak with config:
wandb:     batch_size: 56
wandb:     learning_rate: 0.01237347016736111

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

Training:   0%|          | 0/818 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/235 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: lwtzjows with config:
wandb:     batch_size: 48
wandb:     learning_rate: 0.017067981800936806

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/954 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/274 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/954 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/274 [00:00<?, ?it/s]

VBox(children=(Label(value='0.002 MB of 0.002 MB uploaded\r'),␣
 ↪FloatProgress(value=0.7240437158469946, max=1.0…

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: sirak01v with config:
wandb:     batch_size: 80
wandb:     learning_rate: 0.061346283196273686

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/572 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/165 [00:00<?, ?it/s]

Training:   0%|          | 0/572 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/165 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: cpn70cpf with config:
wandb:     batch_size: 112
wandb:     learning_rate: 0.03758223740619803

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/409 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/118 [00:00<?, ?it/s]

Training:   0%|          | 0/409 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/118 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

wandb: Agent Starting Run: h39mjlnf with config:
wandb:     batch_size: 40
wandb:     learning_rate: 0.02129727952692143

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Epochs:   0%|          | 0/2 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

Training:   0%|          | 0/1145 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/329 [00:00<?, ?it/s]

VBox(children=(Label(value='0.011 MB of 0.011 MB uploaded\r'),␣
 ↪FloatProgress(value=1.0, max=1.0)))

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Total Time: 283.96 min
```

[ ]:
```
# check the last layer input size
number_inputs_last_layer = pretrained_model.fc.in_features

# change the output size of the last layer, but keep the same number of input
pretrained_model.fc = nn.Linear(number_inputs_last_layer, 1)
pretrained_model.fc

best_batch_size = 32
best_learning_rate = 0.01503

model = pretrained_model.to(device)

optimizer = optim.Adam(pretrained_model.parameters(), lr=best_learning_rate)

criterion = nn.MSELoss()
criterion = criterion.to(device)

train_loader, valid_loader, test_loader, train_dataset, valid_dataset,␣
 ↪test_dataset = get_splitted_data(best_batch_size,
                                                                        ␣
 ↪                                 train_data,
                                                                        ␣
 ↪                                 valid_data,
                                                                        ␣
 ↪                                 test_data,
                                                                        ␣
 ↪                                 train_transforms,
                                                                        ␣
 ↪                                 test_transforms)
best_valid_loss = float('inf')
EPOCHS = 3
for epoch in trange(EPOCHS, desc="Epochs"):

    start_time = time.monotonic()

    train_loss = train(model, train_loader, optimizer, criterion, device)
    valid_loss = evaluate(model, valid_loader, criterion, device)

    if valid_loss < best_valid_loss:
        best_valid_loss = valid_loss
        torch.save(model, '/content/drive/MyDrive/ML BigProject/model_CNN.pt')
        torch.save(model.state_dict(), '/content/drive/MyDrive/ML BigProject/
 ↪model_CNN_weights.pt')

    end_time = time.monotonic()
```

```
    epoch_mins, epoch_secs = epoch_time(start_time, end_time)

    print(f'Epoch: {epoch+1:02} | Epoch Time: {epoch_mins}m {epoch_secs}s')
    print(f'\tTrain Loss: {train_loss:.3f}')
    print(f'\t Val. Loss: {valid_loss:.3f}')
```

```
Epochs:   0%|          | 0/3 [00:00<?, ?it/s]

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Using a target size (torch.Size([32])) that is different to the input size
(torch.Size([32, 1])). This will likely lead to incorrect results due to
broadcasting. Please ensure they have the same size.

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

Using a target size (torch.Size([5])) that is different to the input size
(torch.Size([5, 1])). This will likely lead to incorrect results due to
broadcasting. Please ensure they have the same size.

Epoch: 01 | Epoch Time: 4m 3s
        Train Loss: 16524.838
         Val. Loss: 10155.850

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

Epoch: 02 | Epoch Time: 3m 24s
        Train Loss: 15945.919
         Val. Loss: 10176.114

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

Epoch: 03 | Epoch Time: 3m 23s
        Train Loss: 15935.989
         Val. Loss: 10159.310
```

[ ]:
```
                        # check the last layer input size
number_inputs_last_layer = pretrained_model.fc.in_features

# change the output size of the last layer, but keep the same number of input
pretrained_model.fc = nn.Linear(number_inputs_last_layer, 1)
pretrained_model.fc

best_batch_size = 32
best_learning_rate = 0.01503

model = pretrained_model.to(device)

optimizer = optim.Adam(pretrained_model.parameters(), lr=best_learning_rate)
```

```
criterion = nn.MSELoss()
criterion = criterion.to(device)

loss_eval_1 = []
loss_train_1 = []




train_loader, valid_loader, test_loader, train_dataset, valid_dataset,␣
 ↪test_dataset = get_splitted_data(best_batch_size,
                                                                        ␣
 ↪                                train_data,
                                                                        ␣
 ↪                                valid_data,
                                                                        ␣
 ↪                                test_data,
                                                                        ␣
 ↪                                train_transforms,
                                                                        ␣
 ↪      test_transforms)
best_valid_loss = float('inf')
EPOCHS = 6
for epoch in trange(EPOCHS, desc="Epochs"):

    start_time = time.monotonic()

    train_loss = train(model, train_loader, optimizer, criterion, device)
    valid_loss = evaluate(model, valid_loader, criterion, device)
    loss_eval_1.append(valid_loss)
    loss_train_1.append(train_loss)
    if valid_loss < best_valid_loss:
        best_valid_loss = valid_loss
        torch.save(model, '/content/drive/MyDrive/ML BigProject/model_CNN.pt')
        torch.save(model.state_dict(), '/content/drive/MyDrive/ML BigProject/
 ↪model_CNN_weights.pt')

    end_time = time.monotonic()

    epoch_mins, epoch_secs = epoch_time(start_time, end_time)

    print(f'Epoch: {epoch+1:02} | Epoch Time: {epoch_mins}m {epoch_secs}s')
    print(f'\tTrain Loss: {train_loss:.3f}')
    print(f'\t Val. Loss: {valid_loss:.3f}')
```

```
Epochs:   0%|          | 0/6 [00:00<?, ?it/s]
```

```
Training:   0%|          | 0/1432 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([32])) that is different to the
input size (torch.Size([32, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/torch/nn/modules/loss.py:535:
UserWarning: Using a target size (torch.Size([5])) that is different to the
input size (torch.Size([5, 1])). This will likely lead to incorrect results due
to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)

Epoch: 01 | Epoch Time: 3m 23s
        Train Loss: 16973.065
         Val. Loss: 10176.676

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

Epoch: 02 | Epoch Time: 2m 32s
        Train Loss: 15936.145
         Val. Loss: 10173.817

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

Epoch: 03 | Epoch Time: 2m 32s
        Train Loss: 15933.924
         Val. Loss: 10164.567

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

Epoch: 04 | Epoch Time: 2m 32s
        Train Loss: 15945.828
         Val. Loss: 10207.819

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]

Epoch: 05 | Epoch Time: 2m 33s
        Train Loss: 15938.065
         Val. Loss: 10179.009

Training:   0%|          | 0/1432 [00:00<?, ?it/s]

Evaluating:   0%|          | 0/411 [00:00<?, ?it/s]
```

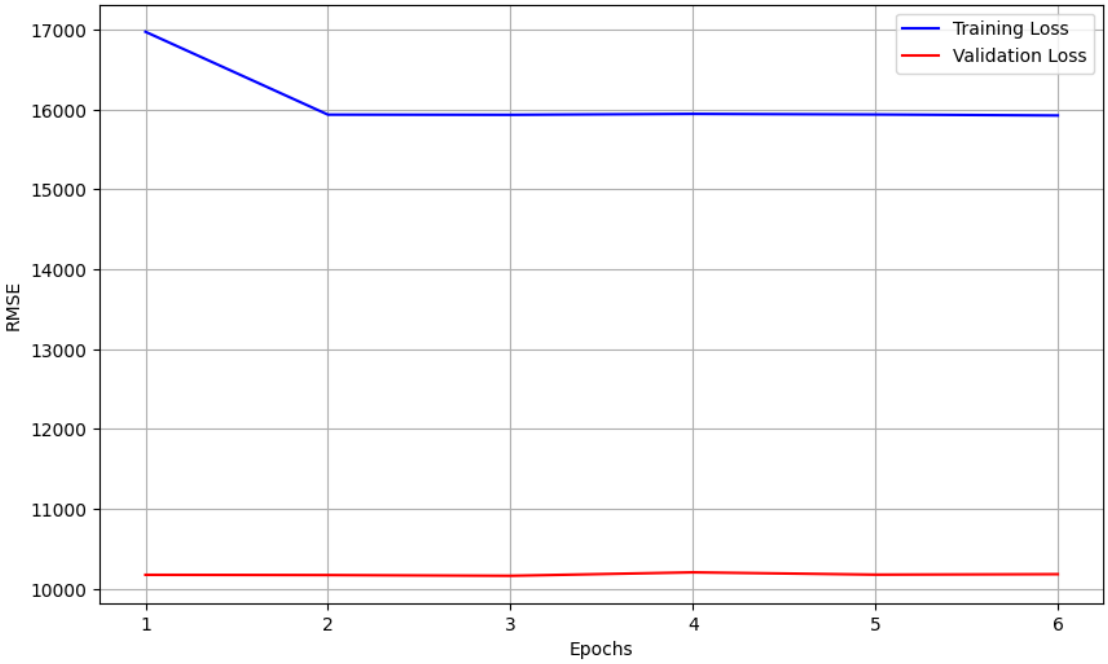
```
Epoch: 06 | Epoch Time: 2m 32s
        Train Loss: 15925.828
         Val. Loss: 10184.630
```

6.2.4 Test of the model

[ ]:
```
# Test the model
best_model_CNN = torch.load('/content/drive/MyDrive/ML BigProject/model_CNN.
 ↪pt', map_location=torch.device("cpu"))
test_loss = evaluate(best_model_CNN, test_loader, criterion, device)
print(f"Test Loss: {test_loss:.3f}")
```

```
Evaluating:   0%|          | 0/578 [00:00<?, ?it/s]

Test Loss: 10746.464

Using a target size (torch.Size([16])) that is different to the input size
(torch.Size([16, 1])). This will likely lead to incorrect results due to
broadcasting. Please ensure they have the same size.
```

[ ]:
```
epochs = range(1, len(loss_eval_1) + 1)
# Plotting the training and validation loss of the CNN
plt.figure(figsize=(10, 6))
plt.plot(epochs, loss_train_1, 'b-', label='Training Loss')
plt.plot(epochs, loss_eval_1, 'r-', label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('RMSE')
plt.legend()
plt.grid(True)
plt.show()
```


Training Loss
Validation Loss
RMSE
Epochs
17000
16000
15000
14000
13000
12000
11000
10000
1
2
3
4
5
6